applied sciences

MDPI

Article

Privacy-Preserving, Scalable Blockchain-Based Solution for Monitoring Industrial Infrastructure in the Near Real-Time

Andrzej Sobecki, Stanisław Barański and Julian Szymański *

Department of Electronic, Telecommunication and Informatics, Gdańsk University of Technology, 80-233 Gdańsk, Poland; andrzej.sobecki@pg.edu.pl (A.S.); stanislaw.baranski@pg.edu.pl (S.B.)
* Correspondence: julian.szymanski@eti.pg.edu.pl

Abstract: This paper proposes an improved monitoring and measuring system dedicated to industrial infrastructure. Our model achieves security of data by incorporating cryptographical methods and near real-time access by the use of virtual tree structure over records. The currently available blockchain networks are not very well adapted to tasks related to the continuous monitoring of the parameters of industrial installations. In the database systems delivered by default (the so-called world state), only the resultant or the last value recorded by the IoT device is stored. Effective use of measurement values recorded in the past requires each time viewing the entire chain of recorded events for a given IoT device. The solution proposed in the article introduces the concept of dependent wallets, the purpose of which is the aggregation and indexation of changes in machine parameters, recorded in the original wallets. As a result, we can easily get data from a certain sensor or sensors in the specified date range, even if the chain of transactions is very long. Our contribution is a universal mechanism that improves the efficiency of the infrastructure monitoring process, which uses blockchains to record measurements from sensors. The proposed model has been experimentally tested on two types of blockchains: Stellar and Hyperledger Fabric.

Keywords: blockchain; Hyperledger Fabric; Stellar; IoT

Citation: Sobecki, A.; Barański, S.; Szymański, J. Privacy-Preserving, Scalable Blockchain-Based Solution for Monitoring Industrial Infrastructure in the Near Real-Time. *Appl. Sci.* **2022**, *12*, 7143. https://doi.org/10.3390/app12147143

Academic Editor: Gianluca Lax

Received: 8 June 2022
Accepted: 8 July 2022
Published: 15 July 2022

1. Introduction

Currently, IoT devices are widely used in the industry and in many personal use cases (e.g., medical equipment, intelligent houses and cars). In each of these areas, we observe a trend of replacing old measurement devices with the new that have additional software and special communication capabilities. Moreover, IoT devices are used in the process of treating and monitoring patients. In certain situations, the collected data may be sensitive or even decide about human life, e.g., insulin pump control, pacemaker calibration, automatic dosing of drugs to patients, drug production process control, etc. In each of them, data consistency and immutability should be guaranteed.

This leads to an increase in the importance of data protection and security. The main problem with acquiring data from those sensors is how we will store the data and ensure that they will not be modified or deleted. This is especially true when the data may contain sensitive data that can be used as evidence in a court case. The currently available solutions are mostly based on the relational or non-relational databases in which data can be easily changed by any administrator.

On the other hand, we have immutable registers based on a blockchain solution. They offer immutability of data based on advanced cryptography. Unfortunately, this technology has huge drawbacks that may not be visible in a small scale project that does not require minimizing delays. When we want to monitor a set of devices in the near real-time (manufacturing process, patients in hospitals) we need to solve the efficiency problems. Many of the existing blockchain networks are slow or very expensive.

Using the IoT, we need to have access not only to the latest data but also to historical records. There are many solutions that offer huge analysis capabilities, but they blindly

trust that the provided data has not been tampered with. There is a lack of tools enabling quick validation of the content and provenance of the data collected by the IoT devices. This can lead to false analytical results as the input data is not verified. We should consider usage of the blockchain networks as immutable registries of data.

Yet again, the problem is the efficient use of data stored in the blockchain, especially when the length of the chain and the unidirectional data structure force us to iterate over each element from the first or last one added to the chain.

In this paper, we focused on improving the efficiency of the blockchain network as a tool for confirming the truthfulness of data analyzed in an external system. The proposed model is also dedicated to supporting monitoring large-scale IoT installation by using the data collected in the blockchain to faster measure metrics (e.g., failure rate, estimated completion time) that require more than only the last measured value.

The main contribution of this paper is a set of improvements based on creating a virtual tree structure using data dependencies between many wallets. This significantly reduces the time needed to find data based on sensor type or date range. The proposed extension can be used in existing blockchain networks, in particular in the area of critical infrastructure monitoring and applications requiring quick access to historical data stored in chains. Moreover, the proposed improvements allow using blockchain in larger projects that require faster analysis based on both current and historical data stored in blockchains.

The paper is constructed as follows. Section 2 contains a discussion regarding the usage of the blockchain to monitor IoT devices. Next, Section 3 contains a problem statement in the context of the usage of blockchain technology in industrial IoT devices (IIoT), especially while we require access to historical data or better performance of systems using blockchain. To achieve the outlined goals, we propose a method of keeping track of logs produced by IoT devices in the blockchain. Section 4 describes the proposed optimizations that allow us to retrieve data from the blockchain in near real-time. The proposed method is generic enough to apply to different types of blockchain, including those that do not support smart contracts. To demonstrate the effectiveness and universality of the proposed mechanism, two prototypes are created for the Stellar network [1] and Hyperledger Fabric [2]. In Section 5, we present the conducted experiments and achieved results. As an example, we prepare and execute a set of queries and their optimized counterparts. Finally, in this section, we discuss issues related to privacy and efficiency when monitoring IoT infrastructure using the blockchain. In Section 6, we present conclusions and potential improvements. We conclude with open issues discussion and future works. As a result of this article, we propose a fully decentralized, privacy-preserving and efficient platform for aggregating and analyzing data using the blockchain network.

2. Blockchain in the Industry

The blockchain-based platforms have been proposed for use in Industry 4.0 as a solution which offers decentralization [3,4], immutability of data [5–7], security [8–10], tracking [11–13], anonymity [14–16], and transparency [17–19].

Moreover, Lu in [20] describes how the blockchain may be useful in the oil industry. The presented example shows how distributed and complicated is the supply-chain and how much data should be integrated to perform the decision-making process. The main problems concern the integration of data, data reliability, and data provenance. In large, distributed companies, delays in updating data between individual units and the central unit should be taken into account. Making decisions in conditions of uncertainty about whether the data is up-to-date can have costly consequences. The authors of the paper suggest that the decision-making process could be improved by the usage of smart contracts. The data stored in the blockchain may improve compliance with the law and increase sharing of data to the government units by using an additional framework such as Trust::DATA (MIT). The authors provide the information that about 3/4 of oil companies were attacked by cybercriminals. It was indicated that the weakest point of the infrastructure is the sensors because based on the generated information, industrial espionage is possible. Abu Dhabi

National Oil Company (ADNOC) uses software developed by the IBM company for the whole lifecycle of oil and gas. This results in reduced drilling time and reduces cost by about 30%. One of the solutions proposed in the paper is the usage of the blockchain, but the authors focused only on the decentralization aspects. To solve the problem with reliability and provenance of data, many oil companies have started using the blockchain in their infrastructure. For example, Sinochem Group (2017) use it to perform electronic transactions for importing oil from the Middle East.

Bodke in [21] tries to summarize all existing solutions that use blockchain and may be useful in modern companies called Industry 4.0. The authors describe dividing platforms and tools into categories that depend on the type of enterprise: medical, smart city, energy industry, financial/business, supply chains/logistics and IoT/manufacturing. The detailed analysis contained in the paper provides the observation that the blockchain platforms are most often used to share data and perform computations rather than transfer assets.

An example of a healthcare solution is HDG (Healthcare Data Gateway) [22], which is created by multiple layers. The first layer is responsible for the secure storage of medical data, the second layer supports processing stored data, and the last layer offers tools for the management of data. Doctors generally use only the processing layer, in which they can view and analyze patients' data. The management layer facilitates the transfer of data and metadata between different medical units and their blockchain platforms through the possibility of integrating two HDGs. The proposed model assumes that the patient has control of his data and that the computation process performed by third-party companies may be performed without a decrease in the privacy of the data. The problem of exchanging the healthcare data is also described in [23], but here, the blockchain is used only as a platform to allow sharing the data collected on the platform. Some improvements for the exchange of patients' data were proposed in [24]. Blockchain was used with smart contracts to manage data registered in external platforms and permissions given to the data. The patient can decide to approve or decline the request for access to his data. In [25], the authors proposed improvements to authorization schemes concerning electronic medical records. They proposed dividing different types of data and different types of queries. In each step, the authors proposed to encrypt data and authorize the user before he can read them.

In [26], the authors described the usage of the blockchain to integrate devices in a smart city and perform all transactions using the blockchain database to improve security. The blockchain was integrated into the communication layer. For each type of service (smart parking, smart cleaning, smart home, etc.), the authors proposed using different ledgers to improve the efficiency of the system.

The next example of smart city service is a lottery system described in [27]. These authors described a blockchain-based architecture which improved the transparency and fairness of a lottery system. Moreover, it was possible to ensure the privacy of transactions in that systems. Finally, the authors created the Fairlotto system, which was based on Ethereum.

In [28], the authors describe how the blockchain was deployed in a pharmaceutical company to support the serialization process. The serialization process was introduced to prepare the traceability of medications. A unique number was assigned to the drugs unit box, case box and unit-load package of drugs, and finally, all these numbers were registered in a central repository (e.g., European Medicines Verification Organisation). After that, in each phase of the packaging and distribution processes, they could add additional information and track changes in the product's location. Due to the regulations, all companies involved in the distribution of drugs have to maintain collected data for a few years. Blockchain was used to achieve shared, decentralized data storage that was resistant to tampering and fraud. Blockchain was integrated with SIFI's SAP software and AntaresVision's real-time control vision devices. Finally, the blockchain stored data about packaging and shipment details with references to the unique drug number.

Another example of a solution dedicated to supporting supply chains is BigchainDB [29]. It is a hybrid solution that incorporates the advantages of blockchain and distributed databases. The authors proposed to create a system that will offer high throughput and low latency with the immutability of data and decentralized controls. The usage scenario of the proposed system was controlling the food production and distribution processes. Several stages of food production and distribution should be monitored to ensure compliance with HAACP standards. The authors indicated 9 levels of control and potential data sources that should be monitored and recorded.

3. Problem Statement

The same functionalities should be offered by service providers who integrate data from IoT devices that are installed at the end-user, such as in smart homes, smart vehicles, power plants, wearable devices, medical devices, industrial, or even military applications. Many of them need an additional level of privacy and security, and they cannot depend solely on mechanisms in centralized databases, such as MariaDB, MongoDB, PostgreSQL, etc. This is mainly because each of these solutions is controlled by the IT department, which has unlimited access to the data. Data are protected as efficiently as the team that protects it is. Such a solution does not guarantee a proper and uniform level of provision of data collection and processing services. The identified problems become more apparent in the case of specific applications.

3.1. Medical IoT Devices

Medical IoT equipment may be wearable or implanted in the human body. They generate sensitive data that could be used to monitor a patient, and based on the computed results, the device may be adjusted. This could be a pacemaker, electrodes for deep brain stimulation, insulin pumps, etc. How do ensure the immutability and non-reproduction of data and guarantee their correct storage and processing in the infrastructure that the client does not know?

Data sent from the device to the servers should be only accessible for certain people or services in a limited range. This data should be read-only because we may be based on it and generate some instruction seta for specific medical devices. If the decision is incorrect, then data have to be immutable, even for the IT department that is responsible for the infrastructure. The court should receive the original data to be able to decide who is at fault: the transmitting device, the algorithm or the doctor. Usually, the team responsible for preparing the analysis, decision models or analytics services is composed of non-medical staff.

While we organize new methods for registering and storing data, we should also note that the data may be a potential source for training and testing ML/AI algorithms. Detailed information about the source of data should be automatically separated from the informative part to preserve the anonymity of the end-user. The provided infrastructure should natively support the separation of personal data from measured values and provide combined information only in strictly defined situations. Data-sharing decisions should also be monitored and recorded in the system.

3.2. Industrial IoT Devices

Currently, the decentralized infrastructure created in enterprise integrates systems in different company suites as well as systems that are provided by external providers. This requires additional mechanisms from the used solutions to control the data on both sides while they are produced and while they are consumed. This is very important, especially when using data from IoT devices to automate production or distribution processes. If the company wants to make decisions based on the data it receives, it must be sure that the source data is correct. Moreover, if such data is crucial to obtain or maintain a competitive advantage, the process of their distribution must be under special control.

This is particularly true if the data describes processes regulated by law and may be the subject of audits or may directly affect the health of customers.

The pharmaceutical factory is an example of an environment which needs private and secure data registered from devices. We can define two main processes that are executed and need decentralized and secure repositories:

- The preparation of the active ingredient of the drug;
- The packaging of the finished product.

In the first area, we should be monitoring the whole process in which chemical compounds are combined to form the active substance. A slight change in the process parameters may result in the production of an active substance with inappropriate properties. Different people are responsible for each production phase, but they also may differ depending on the type of product. Only some subset of persons should have access to the data, and no one should modify the data. If a person wants to steal the collected data, they will be able to read only those data coming from the area for which they are responsible. Problems caused by the human factor in the drug manufacturing process can have a huge impact on potential clients and environments. If that occurs, we should have a valid, immutable data registry in which we can easily trace back and find the reason for the failure. Even if the IT department may want to help someone by erasing or modifying data, this should be impossible.

The packaging area is crucial in preventing trade in counterfeit medicines. Each part of the drug is labelled with a unique number, which allows us to trace the path of the drug from manufacturer to end-user. Coordinating and tracing a drug from manufacturing to end-user requires a unified way to store, control and share data. Another problem concerns people involved in the production process that are rewarded for the trouble-free operation time of the machines, excluding the time needed for calibration and setting up. In real life, they may artificially extend the time of calibration and introduction of new settings. Finally, the time when drugs are produced is limited, and employees have more free time. On the other side, the KPIs are defined well below the realistic level of machine availability. The second problem concerns real failures that are the results of human error.

3.3. Orchestration Based on IoT Data

Smart cities [30] or production plants 4.0 [31–33] are environments in which automation and automatic orchestration is crucial. They need to orchestrate services or devices based on a secure and trusted source of data.

Each decision is made based on data from other devices and control parameters stored in scripts and programs. A more complicated scenario assumes that devices need to coordinate actions using choreography [34] in which we do not distinguish the central coordinator service.

Smart cities face many challenges related to ensuring data privacy [35,36] and the anonymity [37] of measurements or registration of decisions made. This is especially important when integrating heterogeneous data providers to ensure the native independence of a coherent system of entitlements.

Classical platforms [38,39] for gathering events are mainly based on SQL databases and distributed file systems (Apache HDFS, Amazon S3). If the data should be immutable then we usually use some blockchain network (e.g., Hyperledger Fabric, R3 Corda, private network of Ethereum). Improvements based on SQL databases have the same weak point, which is a central point of privilege management.

On the other hand, we have decentralized solutions [40,41] based on popular blockchain networks. Those systems usually struggle with low efficiency or do not have native mechanisms that separate information about the source from reading metric values. By default, the architecture offers full transparency of the transactions performed.

Finally, data stored in the blockchain increases over time, and if we want to have insight into some metrics stored in the data, then the process becomes very slow. To find some data registered by a specific device in some period, we have to analyze each branch—reading

block after block, from leaf to root. For a large-scale environment, this quickly becomes an infeasible approach.

3.4. Summary of Requirements for Industrial Repositories

Based on the performed analysis, we assumed that solutions used for data aggregation and distribution in untrusted, decentralized industrial environments should offer the following functionalities:

- A standardized way of collecting data from edge devices—all systems that collect data from edge devices and other sources should use the same mechanism to register data in a decentralized repository;
- Controlling the provenance of data—metadata about the source of data should be automatically embedded in the structure containing data;
- Providing encryption of transmitted data—transmission of data from the source to the repository has to be secured by the usage of encryption;
- Use of dedicated encryption keys for data related to a specific project (or patient)—in the event of theft of encryption keys or an error in granting access, any possible data leakage should be limited to a narrow range of data, without the possibility of reading other data;
- Immutability of collected data—registered data cannot be changed (even by administrators) and may be only read by privileged users. A company should have a warranty that the data are the same as when they were generated;
- Data are resistant to timestamp tampering—information about the time when data were registered in the repository cannot be changed;
- No data erasure possible—the data cannot be deleted after registration in the repository, and this is not a functionality guaranteed by the application software, but a built-in feature of the data storage and signing mechanism;
- Automatic indexing and aggregating data with metadata for further auditing and control—to enable the online controlling process of other analytic solutions used by the company, the data registered in a secure, decentralized repository should be automatically indexed and aggregated. We should have the ability to quickly compare the basic aggregated values;
- Anti-theft decentralization—the data are saved in such a way that the theft of any of the servers prevents decryption and access to the data stored on it;
- Decentralized consensus for sharing data—the software enables supervised and auditable management of access levels to specific subsets of data;
- Unified data sharing—data may be securely distributed along with the whole enterprise.

4. Methods

To achieve the mentioned requirements for industrial repositories, we prepared a model for incorporating existing systems with the blockchain. In the model, we assumed that the proposed solution can improve systems used in companies if they do not comply with the mentioned requirements.

In this section, we present a method of keeping track of logs produced by IoT devices using a blockchain model which offers collected data for other analytical systems in a faster and securely manner. We decided to use the blockchain because it naively offers most of the required features, such as

- Data replication across all nodes in the network;
- Data immutability;
- Data ownership decentralization;
- High availability;
- Security (authentication, authorization and integrity);
- Fault tolerance (including Byzantine fault).

Dedicated blockchain-based databases such as BigchainDB [42], ProvenDB [43], and LedgerDB [44] provide high transaction rates, low latency, and querying of structured data. They are distributed databases enhanced by some of the blockchain features, such as data integrity, immutability, byzantine-fault tolerance, and access control.

They usually differ in the type of database and consensus protocol; for example, BigchainDB and ProvenDB use MongoDB, LedgerDB is a database on its own, and ChainifyDB [45] offers an overlay layer that can be deployed on any data management system (turning any database into blockchain).

However, in this paper, we present a universal mechanism that can be applied to most blockchain platforms.

4.1. Main Elements

On a general level, we can distinguish three types of cooperated elements:

1. Sensors—IoT devices producing logs and sending them to nodes;
2. Nodes—servers that are receiving signals from sensors, co-creating blockchain networks, and storing logs in immutable form;
3. Clients—systems or users querying nodes for individual data or aggregated results stored in nodes.

Data flow starts with a sensor reading its state, encrypting it, and wrapping it in a blockchain transaction. The sensor then sends the transaction to some available node. This node broadcasts the transaction across the peer-to-peer network, and upon a successful consensus round, updates its state. The transaction is then saved in an immutable repository based on blockchain.

Depending on the implementation, the blockchain is stored in files or some kind of database. In both cases, querying such a big data set is time-consuming. We need to consider the case where we collect a stream of hundreds of events and then want to effectively offer them to others. Therefore, to achieve real-time queries, we need to employ some optimization techniques that we present in our model.

We propose to change classical one-way chain data organization into a time-based tree hierarchy aggregation scheme combined with a computed pattern to achieve efficient querying. A time-based tree hierarchy means that we are creating an overlay tree structure that can be easily indexed by time periods. The computed pattern means that we compute results ahead of query time and store them back in the blockchain, replacing the computation overhead with a storage overhead by keeping all the pre-computed values in the database.

We perform computation at the same moment when new data are saved into the blockchain. Computation is performed in different sliding windows for specified aggregation functions. In the result, if the user needs to compare, e.g., the maximum value for a day, he can query our repository for a pre-computed value and compare it with the value computed in another system. In classical solutions based on blockchain, we would need to query the whole blockchain to find the maximum or average value. What we do instead is to pre-compute these queries for each time frame and store the aggregated values in the blockchain state. As a result, the query can be quickly retrieved without the need to fetch the whole blockchain history.

Implementation of the aggregation process depends on the capabilities of a blockchain platform. Platforms equipped with smart-contract capabilities, a.k.a programmable transactions, can perform aggregation as a part of the storing-log-transaction, whereas platforms without smart-contract capabilities have to introduce a component responsible for producing aggregated values.

The final structure contains blocks with two types of transactions: original data from IoT and additional information that summarizes data from IoT stored in the same block and that summarizes data that was registered in previous blocks. Each summarization transaction contains also the IDs of the first and the last block that describe the range of blocks whose values were aggregated.

To present how the proposed model can work, we prepare the implementation on two heterogeneous blockchain platforms. The first one uses the Stellar blockchain, and the second one uses Hyperledger Fabric. We decided on these two because besides immutable storage, they have little in common; therefore, the successful application of our method on both of them indicates desirable versatility.

4.2. Fabric Hyperledger

Among the field of permissioned blockchains such as Corda [46], Tendermint [47], Quorum [48], and MultiChain [49], Fabric seems to be the most popular one. It is a smart-contract platform, meaning that a transaction can execute any arbitrary (Turing-complete) code and modify the state of the blockchain.

While the security of permissionless blockchains relies on game theory and the use of cryptocurrency, permissioned blockchains like Fabric avoid this problem by identifying each member—both clients and nodes—of the network and controlling their access to it. This also allows the usage of identity-based consensus algorithms like PBFT, Paxos, or Raft, which are usually more efficient than their permissionless counterparts. Fabric decoupling services (called ordering service) are dedicated merely to ordering transactions, hence releasing the rest of the network from participating in a costly consensus algorithm.

The Fabric smart contract, also called *chaincode*, is an application deployed on a Fabric channel. A channel is an isolated network with its own state and blockchain maintained between a set of nodes. Only nodes being part of the channel are permitted to access the blockchain, achieving confidentiality and privacy. This smart contract is protected by an endorsement policy, which specifies conditions that must be met to invoke such a smart contract. An example endorsement policy would say that a transaction is valid if 2-of-3 nodes produce identical state changes. A state change is a set of key-value writes to the state called a *write set*, and a set of key-version reads from the state called a *read set*.

Transactions are delivered to the ordering service, which batches them into blocks. A block is *cut* once the maximum number of transactions or total size (measured in bytes) is reached, or, alternatively, when a defined amount of time elapses. The block is then broadcast to all peers in the network, which validate it, store it in their local block store (implemented as append-only files) and apply the *write set* to their local states. Each node maintains its state in a key-value store (KVS). Transactions are, in essence, changes to the state.

After executing a smart contract, the resulting *write set* is changing the state. If the *write set* is empty, the transaction can be considered useless, since it did not change the outside world. Currently, Fabric supports two KVSs, LevelDB (default) and CouchDB. Access to the current state is possible via `GetState(key)`, and access to transactions which modified the state is possible via `GetHistoryForKey(key)`. Figure 1 presents the relationship between these components.

Each smart contract application runs in a docker container on every node participating in the channel. At the time of writing, Fabric executes smart contracts written in Java, Go, and Node.js (JavaScript/TypeScript).

Each sensor was associated with a unique identifier that was also its key in the state. Each sensor logs the state by reading its value and wrapping it (with additional meta-data) with the transaction. Such a transaction is sent to a node, which executes it and updates the sensor's state. The sensor's state is a key-value entry in the global state (see Figure 1), where the key is the sensor's identifier and the value is a JSON-encoded object with the following structure:

$$L = (L_{\mathrm{id}}, L_{\mathrm{t}}, L_{\mathrm{v}}, L_{\mathrm{u}})$$

where Log L is a 4-tuple, consisting of the sensor's identifier (L_{id}), read time (L_{t}), read value (L_{v}), and measurement unit (L_{u}).

In addition to updating the sensor's state, the smart contract also updates aggregated states. The aggregated state is an entry that collects aggregated data for a specific sensor and time frame. The aggregated state can be formally written as:

$$A = (A_{\text{sum}}, A_{\text{count}}, A_{\text{min}}, A_{\text{max}})$$

where aggregation A is a 4-tuple, consisting of the sum of values (A_{sum}), record count (A_{count}), minimum value (A_{min}), and maximum value (A_{max}).

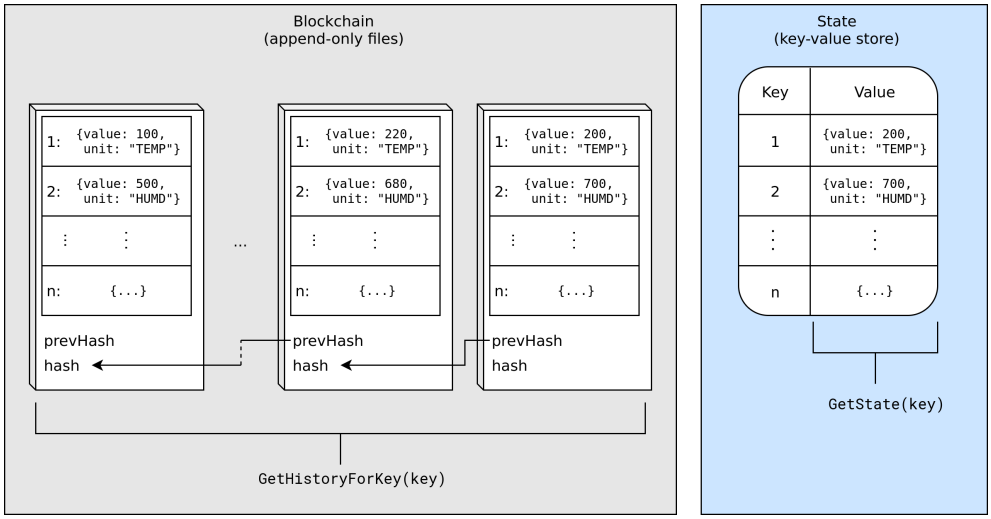

Figure 1. The Fabric's ledger, where the state is determined by sequentially applying transactions stored in the blockchain.

Whenever the smart contract receives a request with argument L, first, it updates the sensor's state, and then it updates aggregated states for each time-frame. A time-frame is created by *cutting off* a part of the RFC3339 formatted log's creation time. For example, for date `2021-08-22T12:34:56`, we create five time-frames:

- Minute—`2021-08-22T12:34`;
- Hour—`2021-08-22T12`;
- Day—`2021-08-22`;
- Month—`2021-08`;
- Year—`2021`.

We then update the aggregated values accordingly to Algorithm 1.

Algorithm 1 Smart-contract's setSensorState function invocation with log L

$\texttt{state}[L_{\text{id}}] = L$
for offset $= 4, 7, 10, 13, 16$ **do**
  $key = \texttt{compKey}(L_{\text{id}}, L_{\text{t}}, \text{offset})$
  $A_{\text{sum}} = \texttt{state}[key]_{\text{sum}} + L_{\text{value}}$
  $A_{\text{count}} = \texttt{state}[key]_{\text{count}} + 1$
  $A_{\text{min}} = \begin{cases} L_{\text{value}} & \text{if } L_{\text{value}} < \texttt{state}[key]_{\text{min}} \\ \texttt{state}[key]_{\text{min}} & \text{otherwise} \end{cases}$
  $A_{\text{max}} = \begin{cases} L_{\text{value}} & \text{if } L_{\text{value}} > \texttt{state}[key]_{\text{max}} \\ \texttt{state}[key]_{\text{max}} & \text{otherwise} \end{cases}$
  $\texttt{state}[key] = (A_{\text{sum}}, A_{\text{count}}, A_{\text{min}}, A_{\text{max}})$
end for

4.3. Stellar

The second blockchain used to demonstrate the proposed improvement is Stellar. Stellar is a blockchain platform dedicated to fast and low-cost payments. Its uniqueness comes from its consensus protocol, Stellar Consensus Protocol (SCP), which is based on Practical Byzantine Fault Tolerance [50] (PBFT) but uses more flexible membership rules. Namely, it does not assume an a priori known list of network participants. The security of the system is achieved by nodes forming a network of trust, which can reflect the real-world relationship between entities.

A stellar network is constructed by nodes running Stellar Core software. The main functionality of this software is performing periodical rounds of Stellar Consensus Protocol [1] with other participants of the network. Stellar Core also lets new nodes (or those which lag behind the current state) catch up with the current state of the ledger. The whole history of transactions is kept in archives, which are stored in flat files not designed for processing or analyzing. For history analysis, a dedicated component is available called Stellar Horizon. This optional component listens for new ledgers published by its companion Stellar Core and stores the whole blockchain in a well-structured and indexed database (PostgreSQL).

Stellar neither supports smart contracts nor the key-value store. Therefore, the aggregation has to be performed by a dedicated component (aggregator) that periodically fetches all transactions within the time frame, computes aggregation functions and publishes results (in the form of regular transactions) back to the blockchain. In our architecture (see Figure 2), the aggregator is a component responsible for listening for new blocks published by the Stellar node, unwinding them to transactions, and decrypting (as described in Section 4.3.1 and calculating aggregation functions such as average, maximum and minimum. Such values are then encrypted and published, in the form of transactions, back to Stellar nodes and stored in the blockchain. Aggregators can produce aggregated values at different time intervals. The minimum is five seconds, which is the interval at which stellar ledgers are created (in private networks, the block creation interval can be adjusted). The minimum interval in each blockchain can differ, and our proposed model may be adjusted.

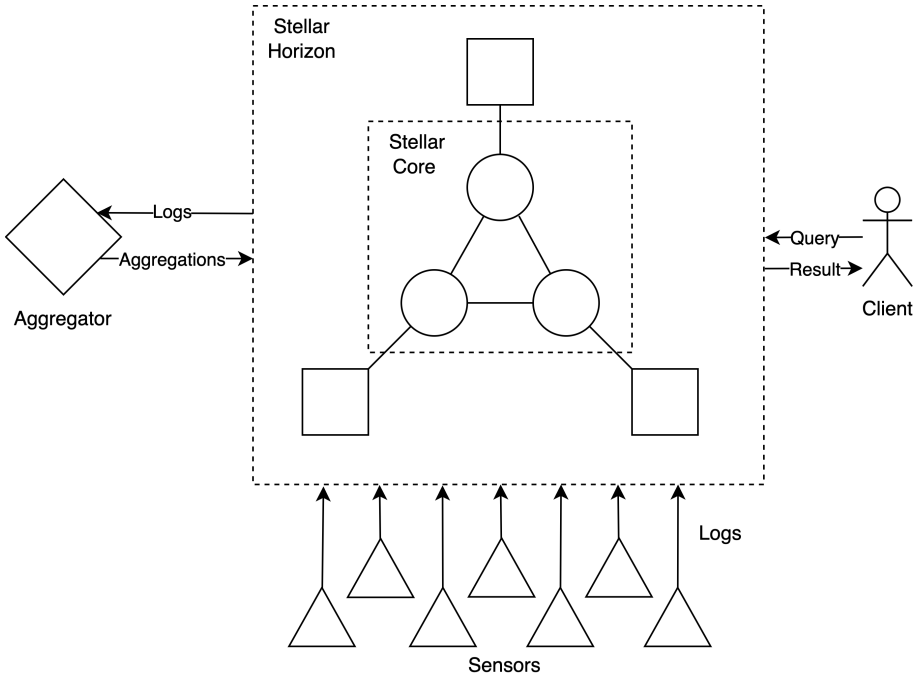

Figure 2. Data flow.

In addition to the aggregated value, the aggregated transaction also contains a span of the transactions being aggregated. This span is encoded by including the first and the last sequence number of the transaction that were aggregated. This allows achieving efficient,

constant access time to the transactions being aggregated. Aggregation details are encoded in the Stellar transactions, as follows:

- Aggregated value—as a `memo` field;
- Time interval—as a `source` account that is associated with the time interval;
- Sensor—as a `destination` account of a payment operation;
- Aggregation function—as a `asset` (or currency) of a payment operation;
- Start of the aggregation span—as the `bumpTo` operation sequence number;
- End of the aggregation span—as the `bumpTo` operation sequence number.

Example structures of blocks with aggregation transactions for one sensor are presented in Figure 3.

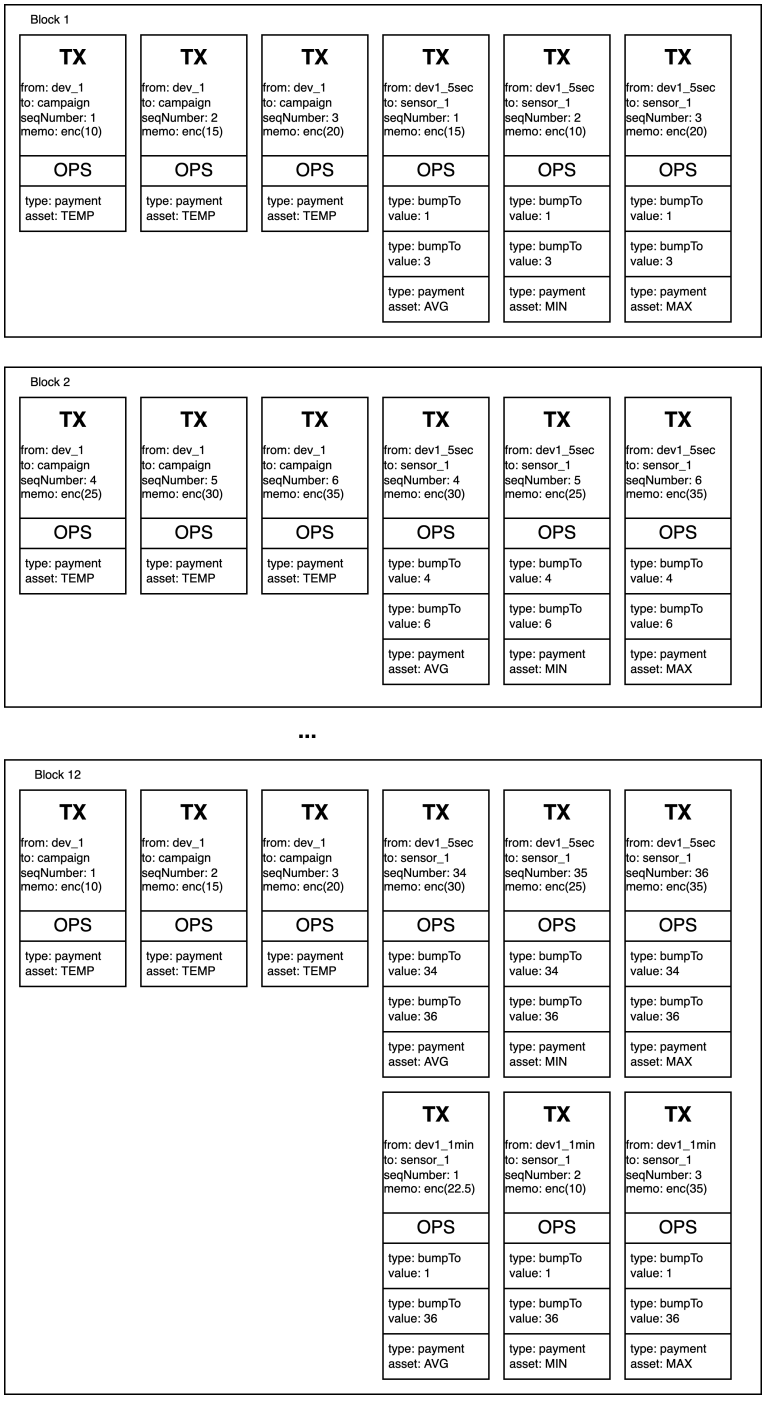

Figure 3. Structure of blocks with aggregation transactions for one sensor.

4.3.1. Encryption

Unlike permissioned blockchains like Fabric, Stellar allows anyone to join the network and replicate the whole blockchain. To achieve data privacy over a public ledger, we propose an encryption mechanism. Sensors' logs are encrypted using asymmetric encryption, meaning each sensor can encrypt only its logs and cannot decrypt any other's. On the other hand, the owner of the campaign (or some other authorized unit) can decrypt all logs. Theoretically, the encryption could be achieved using ElGamal public-key encryption. Each sensor would encrypt a memo field using a publicly known encryption key, while the decryption would be possible only with a corresponding private decryption key. Unfortunately, this scheme produces ciphertext two times larger than the key size. The ciphertext is stored in the memo field, which is bounded to 32 bytes, meaning that keys are bounded to 16 bytes, which are far from secure.

Instead of carrying out public-key encryption, we can leverage the fact that the memo is encoded within the Stellar transaction. Therefore, each transaction is associated with the sender's and receiver's public keys. We can use Elliptic Curve Diffie–Hellman (ECDH) to derive the symmetric encryption key—unique for each sender—and use AES-256-CTR for symmetric encryption. In elliptic curve cryptography, the private key is a n-bit integer where n is the key size.

Let us call the private key k and its corresponding public key represented by the point on the curve K, where $K = k * G$ and G is the known constant point on the curve. If we take two key-pairs, sensor (k_s, K_s) and campaign (k_c, K_c), then using ECDH, we can derive the *shared secret* used to perform symmetric encryption via AES-CTR.

The shared secret can be derived by both

$$k_v * K_b \text{ and } k_b * K_v \tag{1}$$

The keys are equal because

$$k_v * K_b = k_v * k_b * G = k_b * k_v * G = k_b * K_v. \tag{2}$$

There is still one more issue we need to solve. The Stellar protocol uses an ed25519 digital signature scheme based on the Twisted Edwards curve. However, the protocol to achieve ECDH called X25519 uses the Montgomery curve. These curves are birationally equivalent, meaning that we can convert points from one curve to the other [51].

The formula to convert edwards25519 keys to curve25519 keys is as follows:

A private key is generated by taking the first 32 bytes of the hash of the seed, then preventing small-subgroup attacks [52] by clearing the three lowest bits and preventing timing attacks by setting the most significant bit (see Listing 1).

Listing 1. Converting edwards25519 private key to corresponding curve25519 private key. `H` is an SHA-512 hash function, `[:32]` takes the first 32 bytes of the array, `&=` is a bitwise `AND` operator, and `|=` is a bitwise `OR` operator.

```
x = H(seed)[:32]
x[0] &= 0b11111000
x[31] &= 0b01111111
x[31] |= 0b01000000
```

The public key is calculated using the following formula:

$$u = \frac{(1+y)}{(1-y)}$$

where y is the y-coordinate of a point on Edward's twisted curve representing the public key, and u is the resulting public key u-coordinate on the Montgomery curve.

5. Experiments and Results

5.1. Fabric Hyperledger

For our experiment, we used Fabric to provide a test-network configuration that hides the burden of complex PKI management. We decided to write our smart contract in Go and used LevelDB for state storage. A source code used to conduct the experiments is available at https://github.com/stanbar/Blockchain-and-IoT-infrastructure-monitoring (accessed on 7 July 2022).

5.1.1. Environment

Our testing environment was a machine equipped with an Intel i7-4790 (8) 4.000 GHz processor, 16GB (2×8 GB) of DDR3 RAM, and two SSD storages: (1) a Samsung PM85 256 GB dedicated for an operating system and a (2) Samsung 850 EVO 500 GB dedicated for blockchain storage. This machine ran the Arch Linux operating system with kernel version 5.11.8. Docker Version 20.10.5 was used for containerization.

For simplicity, all components ran on one, mentioned above, physical machine, and all sensors were simulated using parallel programs.

We performed a simulation where 50 sensor devices, 2 peer nodes, 1 orderer and 1 client took part. Each sensor generated one log per second; each log was sent to a peer node, which resulted in a total load of 50 transactions per second. The simulation lasted for seven days, which resulted in 604.8 k transactions per sensor or approx 30 M transactions in total. Sensors simulated readings of either temperature (odd identifier) or humidity (even identifier). Temperature reading were simulated using the following formula:

$$\text{temp}(d, h, m, rand) = (10 + \sin(\frac{\pi}{12}(h + \frac{m}{60} - 10))) + \frac{1}{2}\sin(d * \frac{\pi}{3.5}))(1 + \frac{rand - 0.5}{70})$$

Humidity reading were simulated using the following formula:

$$\text{humd}(hour, minute, rand) = (60 + \cos(\frac{\pi}{12}(hour + \frac{minute}{60} - 10) + \pi))(1 + \frac{rand - 0.5}{70}),$$

where d is an integer representing the day of the week in the range $[0, 6]$, h is an integer representing the hour of the day in the range $[0, 23]$, m is an integer representing the minute offset of the hour in the range $[0, 59]$, and $rand$ is a random decimal number in the range $[0, 1)$ and is used to add deviation to the data.

5.1.2. Queries

Query improvements were presented by comparing queries on raw transactions (`GetHistoryForKey`) with queries on aggregated states. Four different queries were used (written in pseudo-SQL).

1. GET ALL TRANSACTIONS WHERE $L_t = time_1$ OR $time_2$ OR $time_3$;
2. GET ALL TRANSACTIONS WHERE $min < L_v < max$;
3. GET AVG WHERE $time_1 < L_t < time_2$;
4. GET COUNT WHERE $L_t < time$.

5.1.3. Results

Table 1 details a comparison of query times with and without aggregation. All four unaggregated queries were executed at a similar time due to the limited nature of the `GetHistoryForKey` API, which always returns all entries in the same order without any filtering; as a result, aggregation functions like avg, count, min, max, cannot be performed until all the records are fetched.

- First and second queries could not be optimized, because all transactions are accessible only via `GetHistoryForKey` API, which does not provide a filtering mechanism; therefore all transactions have to be fetched. The aggregation state does not hold information that could be optimized similar to the Stellar approach. However, this is not possible due to the lack of offset filtering of `GetHistoryForKey` API.
- The execution time of 3rd query was decreased 2244 times. This huge improvement is possible by storing pre-computed averages in the state.
- The fourth query, similarly to the third query, was significantly optimized by storing the total number of records for each time frame in the state.

Table 1. Query times with and without aggregation using Fabric Hyperledger.

Query	w/o Aggregation (s)	w/ Aggregation (s)	Improvement
1	129	N/A	1×
2	130	N/A	1×
3	132.72	0.05914	2244×
4	130.40	0.04998	2609×

5.2. Stellar

5.2.1. Experiments

We performed experiments in a similar configuration as with Hyperledger Fabric. The setup consisted of 50 sensor nodes, 3 Stellar Core nodes each with a Stellar Horizon companion, 1 aggregator, and 1 client.

Each sensor generated one log per second, and each log was sent to a peer node, which resulted in a total load of 50 transactions per second. The simulation lasted for seven days, which resulted in 604.8 k transactions per sensor or approximately 30 M transactions in total.

Sensors simulated readings of either temperature or humidity. For more details, see Section 5.

5.2.2. Queries

Similarly to Hyperledger Fabric, we presented query improvement by comparing queries on raw transactions with queries on aggregated transactions. Here, we also provide the specific SQL queries used to fetch the data from the ProstgreSQL database used by Stellar Horizon.

1. GET ALL TRANSACTIONS WHERE *time* OR *time* OR *time*
   - Aggregation:
     Get aggregation transaction id from specific *time*

```
SELECT transaction_id FROM history_operations ops
JOIN history_transactions txs on ops.transaction_id = txs.id
WHERE source_account = $1
AND type = 1
AND details->>'from' = $2
AND details->>'to' = $3
AND details->>'asset_code' = $4
LIMIT 1
OFFSET $5;
```

     Get the aggregation range

```
SELECT details->>'bump_to' FROM history_operations ops
WHERE transaction_id = $1
AND details->>'bump_to' IS NOT NULL
ORDER BY application_order
```

Get transaction from within the range

```
SELECT memo, account_sequence FROM history_transactions txs
WHERE txs.account = $1
AND account_sequence >= $2
AND account_sequence < $3;
```

- w/o Aggregation

```
SELECT memo, account_sequence FROM history_transactions txs
WHERE txs.account = $1
AND (
(lower(txs.time_bounds) > $2 AND lower(txs.time_bounds) <= $3)
OR (lower(txs.time_bounds) > $4 AND lower(txs.time_bounds) <= $5)
OR (lower(txs.time_bounds) > $6 AND lower(txs.time_bounds) <= $7)
)
```

2. GET ALL TRANSACTIONS WHERE $value_{min} < sensor_n.value < value_{max}$

```
SELECT memo, account_sequence FROM history_operations ops
JOIN history_transactions txs on ops.transaction_id = txs.id
WHERE type = 1
AND details->>'from' = $1
AND details->>'to' = $2
```

3. GET AVG($sensor_n$, *unit*) WHERE *created_at* > time AND *created_at* < *time*
   - Aggregation:

```
SELECT memo, account_sequence, transaction_id FROM
history_operations ops
JOIN history_transactions txs on ops.transaction_id = txs.id
WHERE type = 1
AND details->>'from' = $1
AND details->>'to' = $2
AND details->>'asset_code' = $3
ORDER BY account_sequence DESC
LIMIT $4
```

   - w/o Aggregation

```
SELECT memo, account_sequence FROM history_transactions txs
WHERE txs.account = $1
AND lower(txs.time_bounds) > $2
AND lower(txs.time_bounds) <= $3;
```

4. GET COUNT(*) WHERE SENSOR = $sensor_n$ AND UNIT=*unit* AND *created_at* > *time*
   - Aggregation:

```
SELECT transaction_id FROM history_operations ops
JOIN history_transactions txs on ops.transaction_id = txs.id
WHERE source_account = $1
AND type = 1
AND details->>'from' = $2
AND details->>'to' = $3
AND details->>'asset_code' = $4
ORDER BY account_sequence DESC
LIMIT 1
OFFSET $5;
SELECT details->>'bump_to' FROM history_operations ops
WHERE transaction_id = $1
AND details->>'bump_to' IS NOT NULL
ORDER BY application_order
```

   - w/o Aggregation

```
SELECT count(*) FROM history_transactions txs
WHERE txs.account = $1
AND lower(txs.time_bounds) > $2
GROUP BY txs.account;
```

5.2.3. Results

Table 2 details a comparison of query times with and without aggregation. Reading the logs in plaintext takes approximately 0.723 µs, while reading the same logs in encrypted form takes approximately 60.235 µs, meaning that encryption introduces significant overhead in the form of an approximately 83 times longer reading time.

Table 2. Query times with and without aggregation using Stellar.

Query	w/o Aggregation (s)	w/ Aggregation (s)	Improvement
1	77.859	23.084	3.37×
2	18.552	N/A	1×
3	270.076	72.114	3.75×
4	126.828	75.084	1.69×

6. Conclusions

In the paper, we attempted to improve cooperation between IoT and blockchain technologies. We observed that data collected from IoT sensors are, at most, as secure as we can trust administrators. One of the ways to improve the security of data is using solutions that ensure immutability. Currently, blockchain networks offer needed functionality but do not offer the efficiency needed to support analytics tools, especially when we have to validate the provenance of a bigger dataset containing historical records. In this paper, we proposed an improved method of storing data from IoT sensors in blockchain networks. The original data were extended with metadata that offered additional information about indexes.

We implemented this mechanism for two different blockchains, proving its versatility. For Hyperledger Fabric, we achieved up to 2609 times shorter query time compared to the unaggregated query counterpart, and in Stellar, the improvement was at a level of 3.75. For Hyperledger and any other smart contract platform, the implementation is straightforward, because aggregation can be performed within the smart contract execution. Stellar and other non-smart contract platforms require introducing an external component responsible for performing aggregation updates. For blockchains intended for open networks, the security of the data at rest is assured by storing logs in an encrypted format; therefore, data theft does not reveal any useful information. The proposed optimization technique successfully reduced the time needed to execute aggregation queries on both Stellar and Hyperledger Fabric. The source code used to conduct the experiments is available at https://github.com/stanbar/Blockchain-and-IoT-infrastructure-monitoring (accessed on 7 July 2022).

Further optimization could be achieved by introducing not only time-based indexes but also value-based indexes, allowing optimization of value-based queries, as we have seen in query no. 2. In general, each query can be optimized by introducing indexes. Such an index could be constructed by creating buckets consisting of logs that match a range predicate, e.g., "log's value in range $value_1$-$value_2$", as we showed in this paper with time frames, i.e., "log's time in the range $time_1$-$time_2$".

As we have demonstrated, it is possible to construct such indexes on both transaction-based (Stellar) and smart contract-based (Hyperledger Fabric) blockchain platforms. The proposed model is universal and can be deployed in each blockchain network that exists in industrial infrastructures.

Author Contributions: Conceptualization, J.S.; methodology, A.S.; software, S.B.; validation, A.S.; formal analysis, S.B.; investigation, A.S.; resources, S.B.; writing—original draft preparation, A.S., J.S. and S.B.; writing—review and editing, A.S., J.S. and S.B.; supervision, J.S.; project administration, A.S.; funding acquisition, J.S.; All authors have read and agreed to the published version of the manuscript.

Funding: This work has been supported partially by funds of the faculty of Electronic Telecommunications and Informatics, Gdańsk University of Technology.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Mazieres, D. The stellar consensus protocol: A federated model for internet-level consensus. *Stellar Dev. Found.* **2015**, *32*, 1–45.
2. Androulaki, E.; Barger, A.; Bortnikov, V.; Cachin, C.; Christidis, K.; De Caro, A.; Enyeart, D.; Ferris, C.; Laventman, G.; Manevich, Y.; et al. Hyperledger fabric: A distributed operating system for permissioned blockchains. In Proceedings of the thirteenth EuroSys Conference, Porto, Portugal, 23–26 April 2018; pp. 1–15.
3. Alzahrani, N.; Bulusu, N. Towards true decentralization: A blockchain consensus protocol based on game theory and randomness. In *International Conference on Decision and Game Theory for Security*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 465–485.
4. Chen, Y.; Bellavitis, C. *Decentralized Finance: Blockchain Technology and the Quest for an Open Financial System*; Stevens Institute of Technology School of Business Research: Hoboken, NJ, USA, 2019.
5. Hofmann, F.; Wurster, S.; Ron, E.; Böhmecke-Schwafert, M. The immutability concept of blockchains and benefits of early standardization. In Proceedings of the 2017 ITU Kaleidoscope: Challenges for a Data-Driven Society (ITU K), Nanjing, China, 27–29 November 2017; pp. 1–8.
6. Landerreche, E.; Stevens, M. On immutability of blockchains. In Proceedings of the 1st ERCIM Blockchain Workshop 2018, Amsterdam, The Netherlands, 8–9 May 2018.
7. Ahmad, D.; Lutfiani, N.; Ahmad, A.D.A.R.; Rahardja, U.; Aini, Q. Blockchain Technology Immutability Framework Design in E-Government. *J. Adm. Publik: Public Adm. J.* **2021**, *11*, 32–41. [CrossRef]
8. Khan, M.A.; Salah, K. IoT security: Review, blockchain solutions, and open challenges. *Future Gener. Comput. Syst.* **2018**, *82*, 395–411. [CrossRef]
9. Taylor, P.J.; Dargahi, T.; Dehghantanha, A.; Parizi, R.M.; Choo, K.K.R. A systematic literature review of blockchain cyber security. *Digit. Commun. Netw.* **2020**, *6*, 147–156. [CrossRef]
10. Gupta, A.; Siddiqui, S.T.; Alam, S.; Shuaib, M. Cloud computing security using blockchain. *J. Emerg. Technol. Innov. Res* **2019**, *6*, 791–794.
11. Marbouh, D.; Abbasi, T.; Maasmi, F.; Omar, I.A.; Debe, M.S.; Salah, K.; Jayaraman, R.; Ellahham, S. Blockchain for COVID-19: Review, opportunities, and a trusted tracking system. *Arab. J. Sci. Eng.* **2020**, *45*, 9895–9911. [CrossRef]
12. Sheng, H.; Wang, S.; Zhang, Y.; Yu, D.; Cheng, X.; Lyu, W.; Xiong, Z. Near-online tracking with co-occurrence constraints in blockchain-based edge computing. *IEEE Internet Things J.* **2020**, *8*, 2193–2207. [CrossRef]
13. Neisse, R.; Steri, G.; Nai-Fovino, I. A blockchain-based approach for data accountability and provenance tracking. In Proceedings of the 12th International Conference on Availability, Reliability and Security, Reggio Calabria, Italy, 29 August–1 September 2017; pp. 1–10.
14. Feng, Q.; He, D.; Zeadally, S.; Khan, M.K.; Kumar, N. A survey on privacy protection in blockchain system. *J. Netw. Comput. Appl.* **2019**, *126*, 45–58. [CrossRef]
15. Dorri, A.; Kanhere, S.S.; Jurdak, R.; Gauravaram, P. LSB: A Lightweight Scalable Blockchain for IoT security and anonymity. *J. Parallel Distrib. Comput.* **2019**, *134*, 180–197. [CrossRef]
16. Conoscenti, M.; Vetro, A.; De Martin, J.C. Blockchain for the Internet of Things: A systematic literature review. In Proceedings of the 2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA), Agadir, Morocco, 29 November–2 December 2016; pp. 1–6.
17. Sunny, J.; Undralla, N.; Pillai, V.M. Supply chain transparency through blockchain-based traceability: An overview with demonstration. *Comput. Ind. Eng.* **2020**, *150*, 106895. [CrossRef]
18. Bertino, E.; Kundu, A.; Sura, Z. Data transparency with blockchain and AI ethics. *J. Data Inf. Qual. JDIQ* **2019**, *11*, 1–8. [CrossRef]
19. Chod, J.; Trichakis, N.; Tsoukalas, G.; Aspegren, H.; Weber, M. On the financing benefits of supply chain transparency and blockchain adoption. *Manag. Sci.* **2020**, *66*, 4378–4396. [CrossRef]
20. Lu, H.; Huang, K.; Azimi, M.; Guo, L. Blockchain technology in the oil and gas industry: A review of applications, opportunities, challenges, and risks. *IEEE Access* **2019**, *7*, 41426–41444. [CrossRef]
21. Bodkhe, U.; Tanwar, S.; Parekh, K.; Khanpara, P.; Tyagi, S.; Kumar, N.; Alazab, M. Blockchain for industry 4.0: A comprehensive review. *IEEE Access* **2020**, *8*, 79764–79800. [CrossRef]
22. Jiang, S.; Cao, J.; Wu, H.; Yang, Y.; Ma, M.; He, J. Blochie: A blockchain-based platform for healthcare information exchange. In Proceedings of the 2018 IEEE International Conference on Smart Computing (Smartcomp), Taormina, Italy, 18–20 June 2018; pp. 49–56.
23. Yue, X.; Wang, H.; Jin, D.; Li, M.; Jiang, W. Healthcare data gateways: Found healthcare intelligence on blockchain with novel privacy risk control. *J. Med. Syst.* **2016**, *40*, 1–8. [CrossRef] [PubMed]

24. Theodouli, A.; Arakliotis, S.; Moschou, K.; Votis, K.; Tzovaras, D. On the design of a blockchain-based system to facilitate healthcare data sharing. In Proceedings of the 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), New York, NY, USA, 1–3 August 2018; pp. 1374–1379.
25. Zhang, X.; Poslad, S. Blockchain support for flexible queries with granular access control to electronic medical records (EMR). In Proceedings of the 2018 IEEE International Conference on Communications (ICC), Kansas City, MO, USA, 20–24 May 2018; pp. 1–6.
26. Biswas, K.; Muthukkumarasamy, V. Securing smart cities using blockchain technology. In Proceedings of the 2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Sydney, NSW, Australia, 12–14 December 2016; pp. 1392–1393.
27. Liao, D.Y.; Wang, X. Design of a blockchain-based lottery system for smart cities applications. In Proceedings of the 2017 IEEE 3rd International Conference on Collaboration and Internet Computing (CIC), San Jose, CA, USA, 15–17 October 2017; pp. 275–282.
28. Chiacchio, F.; D'Urso, D.; Compagno, L.; Chiarenza, M.; Velardita, L. Towards a Blockchain Based Traceability Process: A Case Study from Pharma Industry. In *Advances in Production Management Systems*; Production Management for the Factory of the Future; Ameri, F., Stecke, K.E., von Cieminski, G., Kiritsis, D., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 451–457.
29. Tian, F. A supply chain traceability system for food safety based on HACCP, blockchain & Internet of things. In Proceedings of the 2017 International Conference on Service Systems and Service Management, Dalian, China, 16–18 June 2017; pp. 1–6.
30. Chen, L.; Englund, C. Choreographing services for smart cities: Smart traffic demonstration. In Proceedings of the 2017 IEEE 85th Vehicular Technology Conference (VTC Spring), Sydney, NSW, Australia, 4–7 June 2017; pp. 1–5.
31. Pontarolli, R.P.; Bigheti, J.A.; Fernandes, M.M.; Domingues, F.O.; Risso, S.L.; Godoy, E.P. Microservice orchestration for process control in industry 4.0. In Proceedings of the 2020 IEEE International Workshop on Metrology for Industry 4.0 & IoT, Roma, Italy, 3–5 June 2020; pp. 245–249.
32. Liu, B.; Zhang, Y.; Zhang, G.; Zheng, P. Edge-cloud orchestration driven industrial smart product-service systems solution design based on CPS and IIoT. *Adv. Eng. Inform.* **2019**, *42*, 100984. [CrossRef]
33. Tountopoulos, V.; Kavakli, E.; Sakellariou, R. Towards a cloud-based controller for data-driven service orchestration in smart manufacturing. In Proceedings of the 2018 Sixth International Conference on Enterprise Systems (ES), Limassol, Cyprus, 1–2 October 2018; pp. 96–99.
34. Zörrer, H.; Weichhart, G.; Plasch, M.; Vorderwinkler, M.; Kranzer, S.; Prill, D. Chatting roles: A pragmatic service resolution infrastructure for service choreography based on publish/subscribe. *IFAC-PapersOnLine* **2018**, *51*, 1379–1384. [CrossRef]
35. Al-Turjman, F.; Zahmatkesh, H.; Shahroze, R. An overview of security and privacy in smart cities' IoT communications. *Trans. Emerg. Telecommun. Technol.* **2022**, *33*, e3677. [CrossRef]
36. Van Zoonen, L. Privacy concerns in smart cities. *Gov. Inf. Q.* **2016**, *33*, 472–480. [CrossRef]
37. Bouchelaghem, S.; Omar, M. Secure and efficient pseudonymization for privacy-preserving vehicular communications in smart cities. *Comput. Electr. Eng.* **2020**, *82*, 106557. [CrossRef]
38. Fernandez, M.; Jaimunk, J.; Thuraisingham, B. Privacy-preserving architecture for cloud-IoT platforms. In Proceedings of the 2019 IEEE International Conference on Web Services (ICWS), Milan, Italy, 8–13 July 2019; pp. 11–19.
39. Perera, C.; McCormick, C.; Bandara, A.K.; Price, B.A.; Nuseibeh, B. Privacy-by-design framework for assessing internet of things applications and platforms. In Proceedings of the 6th International Conference on the Internet of Things, Stuttgart, Germany, 7–9 November 2016; pp. 83–92.
40. Ali, M.S.; Dolui, K.; Antonelli, F. IoT data privacy via blockchains and IPFS. In Proceedings of the Seventh International Conference on the Internet of Things, Linz, Austria, 22–25 October 2017; pp. 1–7.
41. Lagutin, D.; Bellesini, F.; Bragatto, T.; Cavadenti, A.; Croce, V.; Kortesniemi, Y.; Leligou, H.C.; Oikonomidis, Y.; Polyzos, G.C.; Raveduto, G.; et al. Secure open federation of IoT platforms through interledger technologies-the SOFIE approach. In Proceedings of the 2019 European Conference on Networks and Communications (EuCNC), Valencia, Spain, 18–21 June 2019; pp. 518–522.
42. McConaghy, T.; Marques, R.; Müller, A.; De Jonghe, D.; McConaghy, T.; McMullen, G.; Henderson, R.; Bellemare, S.; Granzotto, A. Bigchaindb: A scalable blockchain database. In *White Paper, BigChainDB*; GmbH: Berlin, Germany, 2016.
43. ProvenDB—Litepaper. 2021. Available online: https://www.provendb.com/litepaper/ (accessed on 1 May 2022).
44. Yang, X.; Zhang, Y.; Wang, S.; Yu, B.; Li, F.; Li, Y.; Yan, W. LedgerDB: A centralized ledger database for universal audit and verification. *Proc. VLDB Endow.* **2020**, *13*, 3138–3151. [CrossRef]
45. Schuhknecht, F.M.; Sharma, A.; Dittrich, J.; Agrawal, D. Chainifydb: How to blockchainify any data management system. *arXiv* **2019**, arXiv:1912.04820.
46. Hearn, M.; Brown, R.G. Corda: A distributed ledger. *Corda Tech. White Pap.* **2016**, *2016*.
47. Tendermint. Tendermint. Available online: https://tendermint.com/ (accessed on 1 May 2022).
48. Quorum. ConsenSys Quorum | ConsenSys. Available online: https://consensys.net/quorum/ (accessed on 1 May 2022).
49. Greenspan, G. Multichain Private Blockchain-White Paper. 2015. Available online: http://www.multichain.com/download/MultiChain-White-Paper.pdf (accessed on 1 May 2022).
50. Castro, M.; Liskov, B. Practical Byzantine fault tolerance. In Proceedings of the OSDI, New Orleans, LA, USA, 22–25 February 1999; Volume 99, pp. 173–186.

51. Langley, A.; Hamburg, M.; Turner, S. RFC 7748—Elliptic Curves for Security. 2016. Available online: https://tools.ietf.org/html/rfc7748 (accessed on 1 May 2022).
52. Zuccherato, R. RFC 2785—Methods for Avoiding the "Small-Subgroup" Attacks on the Diffie-Hellman Key Agreement Method for S/MIME. 2000. Available online: https://tools.ietf.org/html/rfc2785 (accessed on 1 May 2022).