algorithms

Article

A Blockchain-Based Audit Trail Mechanism: Design and Implementation

Cristina Regueiro *, Iñaki Seco , Iván Gutiérrez-Agüero, Borja Urquizu and Jason Mansell

Tecnalia, Basque Research and Technology Alliance (BRTA), Mikeletegi Pasealekua 2, 20009 Donostia-San Sebastián, Spain; inaki.seco@tecnalia.com (I.S.); ivan.gutierrez@tecnalia.com (I.G.-A.); borja.urquizu@tecnalia.com (B.U.); jason.mansell@tecnalia.com (J.M.)
* Correspondence: cristina.regueiro@tecnalia.com

Abstract: Audit logs are a critical component in today's enterprise business systems as they provide several benefits such as records transparency and integrity and security of sensitive information by creating a layer of evidential support. However, current implementations are vulnerable to attacks on data integrity or availability. This paper presents a Blockchain-based audit trail mechanism that leverages the security features of Blockchain to enable secure and reliable audit trails and to address the aforementioned vulnerabilities. The architecture design and specific implementation are described in detail, resulting in a real prototype of a reliable, secure, and user-friendly audit trail mechanism.

Keywords: Blockchain; audit trail; reliability; integrity; usability; monitor

Citation: Regueiro, C.; Seco, I.; Gutiérrez-Agüero, I.; Urquizu, B.; Mansell, J. A Blockchain-Based Audit Trail Mechanism: Design and Implementation. *Algorithms* **2021**, *14*, 341. https://doi.org/10.3390/a14120341

Academic Editor: Shiping Chen

Received: 21 October 2021
Accepted: 24 November 2021
Published: 26 November 2021

1. Introduction

Nowadays, we are in a fully digitalized world where the number of connected devices is growing exponentially due to their great usefulness in various sectors such as advanced manufacturing, wearables, connected vehicles, etc. However, although connected devices may seem harmless, in reality they are susceptible to being accessed and manipulated (due to their Internet connectivity), and they do not often follow good security practices; for example, software updates and patches are relatively difficult to perform [1,2]. This makes them easy prey for cybercriminals looking for such devices as an entry point into corporate networks or other more protected parts of the network. For this reason, several security solutions have been suggested in the literature to ensure the security of connected devices [3–5]; however, this security should still be audited to achieve a high level of confidence in these devices [6].

Well-managed audit trails are currently key indicators of good internal business control, not only with connected devices, but in any type of environment, as they are critical components in providing useful services such as audit processes, secure information storage, tracking of changes made to recorded data, and detection of discrepancies, anomalies, and malicious activities [7,8]. Audit trails have evolved from manual to automated electronic records gaining in accuracy, accessibility, and usability. However, despite their great utility, they are not without vulnerabilities [9,10]. Current audit trail implementations are vulnerable to several attacks, which allow adversaries to manipulate data, meaning that integrity could be severely compromised. Traditional audit data protection schemes, such as those used in some online transaction processing [11], include the use of an append-only device such as continuous feed printer or write once read multiple (WORM) optical devices. These systems operate under the weak security assumption that the logging site cannot be compromised. However, this security assumption is totally inadequate, and some attackers have managed to exploit logging vulnerabilities [12]. Once attackers enter the system, they can corrupt the accessed audit logs or even completely disable the audit functionality.

Moreover, most organizations that include audit functionalities store the trail in relational databases, which can be easily altered or even deleted, leading to a lack of trust in the process. In addition, audit trails are usually under the control of a central authority that controls and manages information records. In this context, the reliance on that central authority to maintain correct and accurate information is reduced because there is no mechanism to verify the status of the audit logs. To counter the aforementioned attacks, Blockchain technology [13] has started to be considered as a suitable technology for auditing purposes. Although the rise of the Blockchain technology is quite recent, it really offers the promise of a secure, transparent, and affordable solution for audit trails. On the one hand, Blockchain solves the problem of trust in a central authority by maintaining records and transactions of information resources across a distributed network, rather than in a central authority. On the other hand, Blockchain creates an immutable record of transactions where nonrepudiation is guaranteed by design. In other words, the need for immutable audit data storage that is not governed by a central authority can be guaranteed with Blockchain. However, although Blockchain has been shown to be a suitable technology to backbone audit trails due to its integrity, decentralization, and nonrepudiation characteristics, it is still not widely deployed as it is not easy to use [14].

The main contribution of this paper is the detailed design and implementation of a real prototype of a Blockchain-based audit trail mechanism to improve the security and usability of current audit trails solutions.

The paper is organized as follows: Section 2 starts with an overview description of Blockchain technology and the main existing Blockchain-based audit mechanisms, identifying their main limitations that have motivated this research. Sections 3–5 present the design, the implementation, and the operation of the proposed audit trail mechanism. Section 6 describes some experiments carried out for its validation. Lastly, Section 7 draws the main conclusions and future work of the research.

2. Background and Motivation

2.1. Blockchain Technology

Blockchain [13] has emerged as an immutable transaction ledger created through a distributed network of peer nodes that maintain a copy of the ledger by applying transactions that have been previously validated by a consensus protocol and grouped into blocks with a cryptographic hash that links each block to the previous one. In this way, given the last block, the previous ones cannot be modified without altering the subsequent blocks (i.e., the data are resistant to modification). Another key aspect of these data structures is that transactions are digitally signed by the creator, so that the origin of a piece of data can always be traced back to its creator. In addition, Blockchain eliminates the need for a central controlling authority to manage transactions or maintain records (disintermediation).

There are different types of Blockchain technologies [15,16]. Bitcoin [17] was the first public Blockchain. It was presented by Satoshi Nakamoto and has been used as a widely available payment system. Later, Ethereum [18], another public Blockchain, appeared with a new and ambitious goal based on enabling distributed computing through smart contracts. This was one of the main improvements of Blockchain technology: decentralized applications (DApps) based on smart contracts that provide the abstraction and execution of business logic on a trusted platform. Smart contracts [19] are self-executing and self-enforcing computer contracts that are governed by explicit terms and conditions set between two or more parties.

In addition to public Blockchain networks where there are no restrictions on participation, private Blockchains that are used with authorized participants have also started to appear. A governing entity (a set of participants) usually allows the entry of new participants and decides on their permissions. In this context, Hyperledger Fabric [20] is the most famous private Blockchain network; Ethereum also has its own private version, known as Quorum [21], created by JP Morgan and currently managed by ConsenSys [22]. Both offer privacy, robustness, scalability, and improved performance. However, Quorum stands out for its simplicity and high maturity, as it is based on a fork of the Ethereum Blockchain with the main design goal of reusing existing technology as much as possible and maintaining synchronization with upcoming versions of the public Ethereum codebase.

2.2. Current Blockchain Based Audit Solutions

Although the literature has widely proposed the use of Blockchain technology to solve some of the main vulnerabilities of traditional audit trail mechanisms, as shown in surveys [23–25], there is still limited research on actual implementations that not only consider security improvements, but also take into account a high level of usability.

Several industry blog posts, such as [26,27], as well as articles from the research community [28–30], have scratched the surface on the use of Blockchain for auditing, but without delving into specific implementation details. However, there have been some attempts to realize the idea of Blockchain-based auditing systems. Table 1 summarizes the main contributions of existing approaches, in addition to identifying their main limitations that will be overcame with the proposed method.

Table 1. Blockchain-based audit trail existing approaches (contributions and limitations).

Study	Main Contribution	Main Limitations
[31]	Upgrade of an existing traditional auditing system to improve its security using Blockchain.	Lack of technical aspects specific to the Blockchain technology.
[32]	Detailed description of how to use Blockchain for bank accountability.	Limited to banking context (broader applicability limited). Usability is not considered.
[33–36]	Auditing mechanisms based on data recorded in traditional database/InterPlanetary File System (IPFS) systems/secure log systems + tamperproof in Blockchain.	Availability is at risk (traditional databases). Two systems are required; complexity increases (synchronization is required).
[37]	Detailed description of how to use Blockchain for wide area protection systems	Limited application. Usability is not considered.
[38]	Detailed description of how to use Bitcoin for secure log management.	Auditing functionality is not secure enough (Blockchain only saves data but does not execute auditing functions; an additional log server is needed). Usability is not considered.

While all previous studies on Blockchain-based approaches for auditing mechanisms aimed at improving security, none of them addressed usability aspects, which are rather limited when Blockchain technology comes into play. Today, Blockchain accounts are needed to interact with the Blockchain and to obtain information recorded on it through the logic of DApps. This is often a limiting requirement as it is not currently common for ordinary users to have a Blockchain account and, even if they do, to know how to use it properly.

2.3. Motivation

Motivated by the limitations of existing research about Blockchain-based auditing systems identified in Section 2.2, the aim of this research was to go deeper including the particularities of the Blockchain-based implementation for a general-purpose solution, as well as usability aspects. In this sense, a Blockchain monitoring tool is recommended that provides a graphical and web-based interface to access the information recorded in the Blockchain in a user-friendly way, with the Blockchain being fully transparent to users. In this way, users unfamiliar with the underlying technology will also be able to use the auditing system, enhancing the mass adoption of this solution.

Taking into account above considerations, the main requirements for the general-purpose Blockchain-based audit trail mechanism proposed in this research that enhances the security and usability of current audit trails solutions are gathered in Table 2.

Table 2. Blockchain-based audit trail mechanism requirements.

Requirement Name	Description
Inalterability (integrity)	The audit trail mechanism should provide inalterable evidence of information needed for audit purposes.
Long-term maintenance	The audit trail mechanism must retain evidence of past information (history records).
Assurance of origin	The origin of all data recorded in the audit system should be securely identifiable.
Simultaneity	The audit trail mechanism should be able to support simultaneous input/output interaction from different stakeholders.
Availability	The audit trail mechanism must have a very high availability (close to 99%).
Confidentiality	All incoming and outgoing communications must be encrypted.
Access control	Data recorded in the audit trail mechanism should only be provided and/or consumed by authorized users.
Usability	The audit trail mechanism must provide a graphical and user-friendly interface.
Multidomain	The audit trail mechanism should be used in any kind of use-case, without limiting it to a particular environment.
Abstraction of the Blockchain	The audit trail mechanism must provide the possibility of accessing the recorded data and abstracting the Blockchain technology, thus enabling its mass adoption.
REST API	The audit trail mechanism must provide an REST API to facilitate the input/output interaction of information sources.

3. Audit Trail Mechanism Architecture

Audit trails are information records that catalog important events to provide supporting documentation and history useful for security and operational actions or for mitigating challenges, including details on the date, time, and user information associated with the event. They provide a baseline for audit or analysis when necessary (an error has been detected, risk management, etc.).

The proposed mechanism is implemented through smart contracts backboned by a Blockchain network, to take advantage of the following inherent features of Blockchain:

- Integrity: The information recorded in Blockchain cannot be manipulated. As the information is distributed across multiple nodes, it cannot be modified at all. In this way, fraud prevention and detection are almost unnecessary.
- Authenticity: Blockchain provides an unalterable assurance of origin; thus, nonrepudiation is guaranteed.
- Long-term maintenance of audit information: Once information has been entered into the Blockchain, it can never be deleted; consequently, the audit trail will always be traceable.

Figure 1 shows the architecture design of the proposed Blockchain-based audit trail mechanism.

It consists of five main elements:

- Blockchain network: A permissioned Blockchain network should be considered as the backbone. All information provided to the audit trail will be recorded in all the nodes of the Blockchain, ensuring its integrity, authenticity, and long-term maintenance. This means that both the auditing system functionality and the information registered in the system cannot be modified at all, making them totally reliable.

- Smart contracts: The audit trail functionalities are implemented through smart contracts deployed on the Blockchain network. Like information, smart contracts are registered on all the nodes of the Blockchain, ensuring their integrity and secure execution. Auditing functionalities include the registration of necessary data to be audited on the Blockchain, as well as the consumption of this previously registered data for integrity validation. In addition, Blockchain-based events will be generated to feed the Blockchain monitor.
- Blockchain client: Each component using the auditing functionalities backboned by Blockchain needs a Blockchain client to be able to interact with the Blockchain. This Blockchain client will be deployed as a docker image for usability and interoperability reasons. This Blockchain client exposes a REST API to be able to interact with it.
- Blockchain monitor: It listens for Blockchain events from the audit trail mechanism smart contract and normalizes and categorizes the details for correct consumption from the Blockchain monitor client. The Blockchain monitor allows isolating external users from the need to have a Blockchain client (or, at least, a Blockchain account) to consume the information recorded on the Blockchain.
- Blockchain monitor client: This client consumes the normalized and categorized information from the Blockchain monitor.

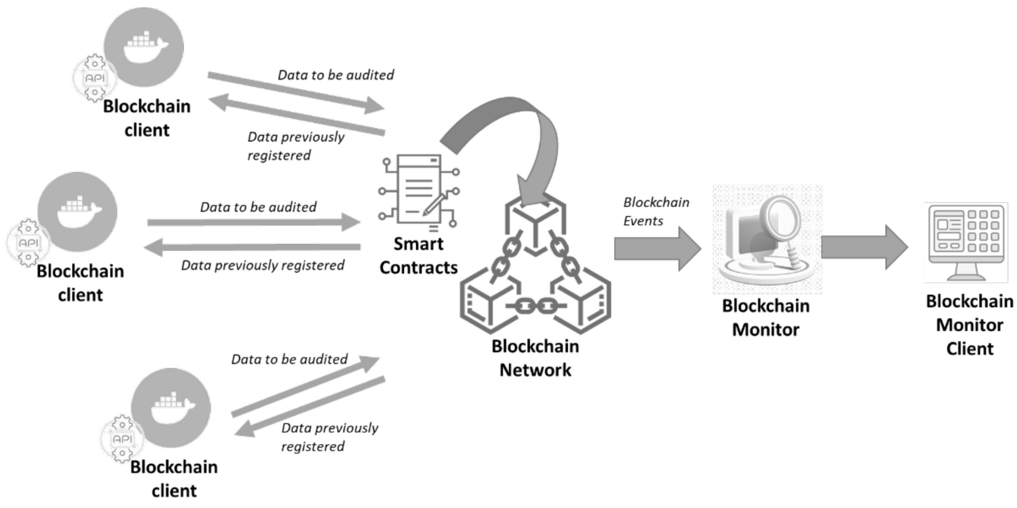


Figure 1. Audit trail mechanism architecture.

4. Audit Trail Mechanism Description

This audit trail mechanism was designed and implemented within the Horizon 2020 project KYKLOS 4.0 (Grant agreement ID: 872570), where it will also be validated to provide audit trail functionalities in the manufacturing ecosystem.

This section describes the specific details of the implementation of each of the components of the audit trail mechanism.

4.1. Blockchain Network

A permissioned Quorum network was considered for the audit trail mechanism due to some of its characteristics:

- Smart contracts are quite advanced. They are based on Solidity [39], allowing the use of all the logic of a complete Turing machine [40].
- Without cryptocurrency. There are several applications, such as audit trails, that do not require cryptocurrencies. In these cases, there is no need to complicate the operation of the network with cryptocurrency support; the revenue model, if necessary, should be outside the network.
- New consensus algorithms. Quorum leverages RAFT [41] for collision fault tolerance and IBFT consensus [42] for Byzantine fault tolerance, increasing speed and not

requiring large processing capacity at the nodes as in, for example, public networks with proof-of-work consensus algorithms [43]. Consequently, performance increases.

- Increased scalability, in terms of both participants and activities. Quorum is a permissioned network in which only authorized nodes can participate. These new participating nodes must have permission to be part of the Blockchain and be able to verify transactions, execute smart contracts, and maintain the state of the ledger. Figure 2 shows the architecture of a Quorum node.

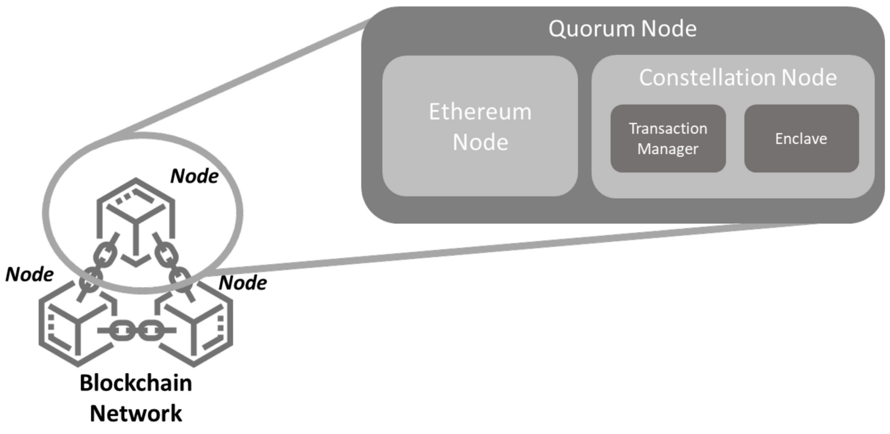


Figure 2. Quorum node architecture.

As introduced in Section 2, the Quorum implementation is a fork of Ethereum, adapting it to work in a permissioned context. Quorum nodes manage the communication with Blockchain clients, as well as with other nodes, using the general-purpose Constellation p2p system to communicate with secure messages [44]. The Constellation module in the Quorum node consists of two submodules:

- Transaction manager: It manages private transactions by allowing access and sending and receiving encrypted payloads to other transaction managers on other Quorum nodes. It uses Enclave to perform cryptographic operations.
- Enclave: It performs all necessary encryption and decryption operations.

4.2. Smart Contracts for Audit Trail

The intelligence for the audit trail mechanism is implemented through smart contracts deployed on the Blockchain network. The main functionalities are described below.

4.2.1. User Management

The possibility to use an audit trail mechanism should be limited to authorized users only. For this purpose, two types of users are defined:

- Administrators: These entities are those who can authorize or deny users to use the audit trail mechanism. Administrators can also define or delete other new administrators. The first administrators in the system are the participants of the Blockchain network.
- Authorized users: These users are authorized by an administrator to use the audit trail mechanism. The audit trail data associated with each user can only be updated or retrieved by its owner (the user who registered it). Therefore, although multiple users may register audit trail information on the same Blockchain network, each user will only have access to the information associated with themself.

Users are identified in the Blockchain-based audit trail mechanism by their Blockchain address (it is unique). No additional personal information is needed.

4.2.2. Administrator Functionalities

Administrators are the only users who can execute functionalities related to the general management of the system. These functionalities include (but are not limited to) the following:

- Define new administrators (the new administrator ID is needed).
- Remove an existing administrator (the administrator ID is needed).
- Check if a specific user ID is an administrator in the system (the administrator ID is needed).
- Obtain the number of administrators in the system.
- Authorize a new user (the new authorized user ID is needed).
- Deauthorize an existing authorized user (the authorized user ID is needed).
- Check if a specific user ID is an authorized user (the authorized user ID is needed).
- Obtain the number of authorized users in the system.

4.2.3. Authorized User Functionalities

Authorized users are those who can use the audit trail mechanism by recording or retrieving the audited data. A specific audit trail data model is needed. The following fields are proposed as an example for the audit trail data model:

- ID: This is an internal ID to identify the audit trail record in the Blockchain. It is automatically generated by the smart contract.
- Owner: This refers to the authorized user ID (authorized user Blockchain address) who has registered the audit trail. The owner is the only one with permission to update or retrieve this information. It is automatically detected by the smart contract.
- Data: This refers to the audit trail registered in the system for auditing purposes. The data could be provided by the authorized user as a name-value, to allow the registration of different types of information and provide a general-purpose solution (not limited to a specific application or domain). The value can be encrypted if necessary.
- creationTimestamp: This refers to the timestamp of the audit trail data registering process in seconds since the epoch. It is automatically generated by the smart contract.
- updatedTimestamp: This refers to the timestamp of the audit trail data last updating process in seconds since the epoch. It is automatically updated by the smart contract.

The functionalities related to the audit trail mechanism for authorized users include (but are not limited to) the following:

- Register a new audit trail (data information is needed).
- Update an existing audit trail (ID of the audit trail and new data are needed).
- Obtain an existing audit trail (ID of the audit trail is needed).

It is not possible to delete information recorded in the Blockchain, which is why the "delete" functionality has not been included.

Every time an operation is executed in the smart contract, a Blockchain-based event is generated to feed the Blockchain monitor. The Blockchain-based events to be generated include (but are not limited to) the following:

- A new administrator is added.
- An existing administrator is removed.
- A new user is authorized.
- An existing user is deauthorized.
- A new audit record is registered.
- An existing audit record is updated.

4.3. Blockchain Client

The Blockchain client is the application that users need to interact with the functionalities of the Blockchain-based audit trail implemented through smart contracts. In this context, the main functionalities of the Blockchain client are described below.

4.3.1. Blockchain Account Management

Each user using the Blockchain-based audit trail mechanism requires a Blockchain account. This consists of a Blockchain address, which identifies the user in the Blockchain network (in fact, it will be the user ID in the system), as well as an associated private key which should only be known by the user (the private key should be securely stored.). The account information is usually managed through a "Blockchain wallet", whose implementation is included in the Blockchain client to isolate users from any Blockchain dependency. In this context, the functionalities available in the Blockchain client related to account management are the following:

- Create a new account. A new address and an associated private key are generated.
- Obtain the address associated with a private key (the private key obtained when creating a new account is needed).
- Add the account to the wallet (the private key is needed).
- Obtain the address added to the account. The address previously added to the wallet is obtained.

In theory, wallets can store information about different accounts. However, for accountability reasons, the number of accounts per wallet is limited to one as the address will be the user identifier (user ID) in the system; thus, only one address should be managed by each wallet. For this reason, as there is only one account per wallet, the Blockchain client automatically uses the account details from the wallet without the need to explicitly provide them.

4.3.2. Blockchain Transactions Generation

The interaction with the smart contract deployed on the Blockchain is done through Blockchain transactions. The Blockchain client automatically generates the transaction necessary to execute all the functionalities available in the smart contracts. The Web3.js library [45] is used.

4.3.3. REST API for External Interaction

The Blockchain client exposes an REST API to allow users to easily use the Blockchain client and execute all functionalities described above. In this way, integration in different systems is easier.

4.4. Blockchain Monitor

The Blockchain monitor listens for Blockchain-based events generated by the smart contracts, notifying about updates in the audit trail. It provides a mechanism for external users (e.g., external auditors) to verify the audit trail recorded on the Blockchain in an easy way (making Blockchain transparent for users).

Figure 3 shows the internal architecture of the implemented Blockchain monitor.

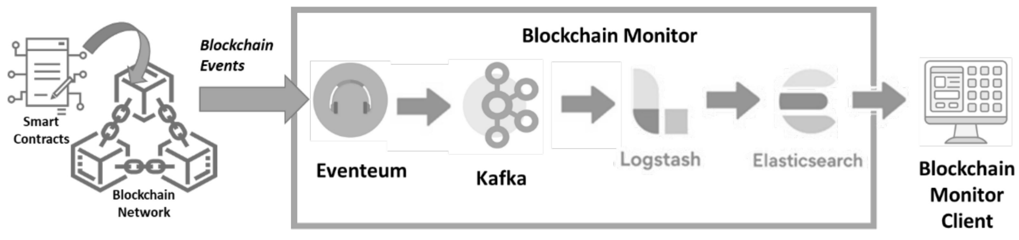


Figure 3. Blockchain monitor architecture.

The Blockchain monitor consists of four main elements:

- Eventeum [46]: This is a middleware that serves as a bridge between the smart contracts deployed on the Blockchain network and the application layer of the Blockchain

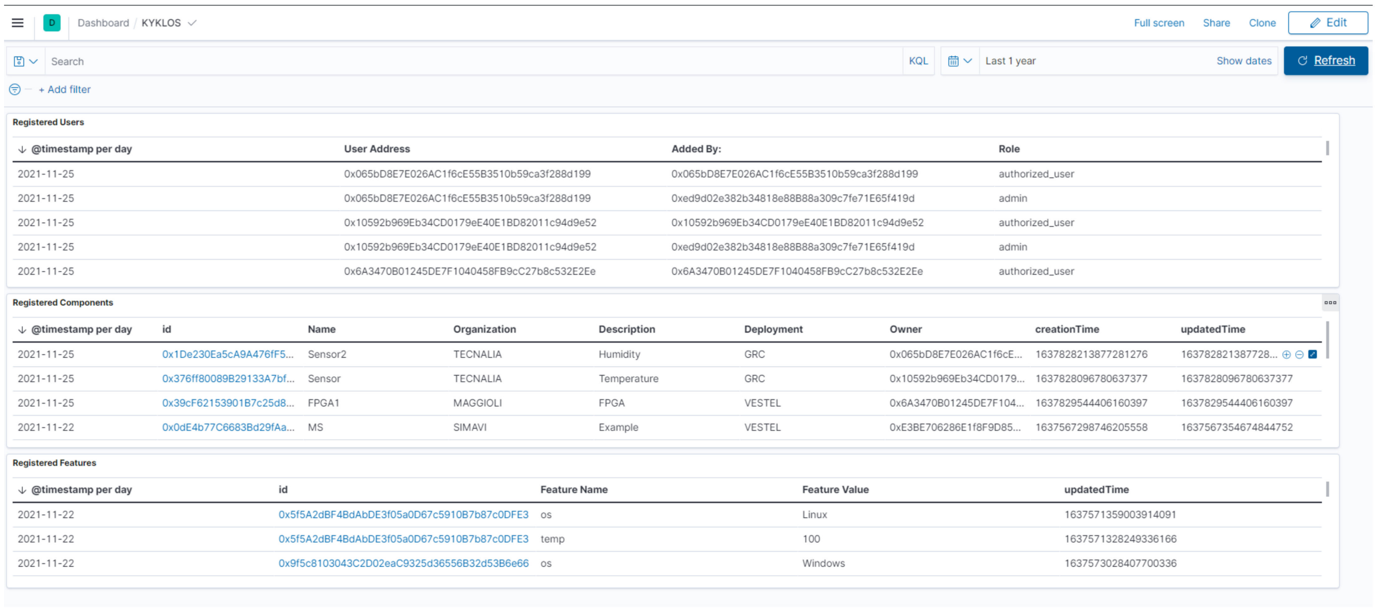
monitor. It is in charge of listening to all the Blockchain events from the Blockchain-based audit trail mechanism and transmitting them to Logstash using Kafka.
- To listen to the events, it is necessary to be subscribed to the events of the specific smart contract addresses (each smart contract has a Blockchain address, which identifies itself in the Blockchain). In addition, it is also necessary to indicate to Eventeum the specific format of the events (event ID, order of the event parameters, type of the event parameters).
- Apache Kafka [47]: This is the intermediary platform used to distribute and process message flows between Eventeum and Logstash. It allows data to be communicated very quickly, categorizing them in different topics if necessary.
- Logstash [48]: This is a log management tool whose function is to collect all the events it receives from Eventeum, through Kafka, and normalize them before routing them to the Elasticsearch service for processing.
- Elasticsearch [49]: This is a distributed search and analysis engine, which allows storing, indexing, and processing information. It receives data from the Logstash service and categorizes the information into different categories: users and data. The information stored in Elasticsearch can be recreated from scratch at any time in case of a security incident, resulting in a fully reliable source of information. Elasticsearch exposes an REST API to access the information.

4.5. Blockchain Monitor Client

There may be several options for consuming data from the Blockchain monitor. However, Kibana [50] was considered the best option due to its high compatibility with Elasticsearch, as well the large number of graphical capabilities it offers, which would greatly improve the usability of the system. Kibana is a graphical interface which displays the information analyzed in Elasticsearch in real time through customized dashboards. Access to Kibana dashboards requires authentication, and different roles can be created to access different types of information.

In the audit trail mechanism, Kibana graphically displays information about the Blockchain events. Two types of dashboards are proposed:

- The general dashboard, where the current audit trail information is displayed (latest updates). Figure 4 shows a general dashboard from the KYKLOS 4.0 project, as an example, showing information about registered users, registered components, and registered features of the different components.



Registered Users

@timestamp per day	User Address	Added By:	Role
2021-11-25	0x065bD8E7E026AC1f6cE55B3510b59ca3f288d199	0x065bD8E7E026AC1f6cE55B3510b59ca3f288d199	authorized_user
2021-11-25	0x065bD8E7E026AC1f6cE55B3510b59ca3f288d199	0xed9d02e382b34818e88B88a309c7fe71E65f419d	admin
2021-11-25	0x10592b969Eb34CD0179eE40E1BD82011c94d9e52	0x10592b969Eb34CD0179eE40E1BD82011c94d9e52	authorized_user
2021-11-25	0x10592b969Eb34CD0179eE40E1BD82011c94d9e52	0xed9d02e382b34818e88B88a309c7fe71E65f419d	admin
2021-11-25	0x6A3470B01245DE7F1040458FB9cC27b8c532E2Ee	0x6A3470B01245DE7F1040458FB9cC27b8c532E2Ee	authorized_user

Registered Components

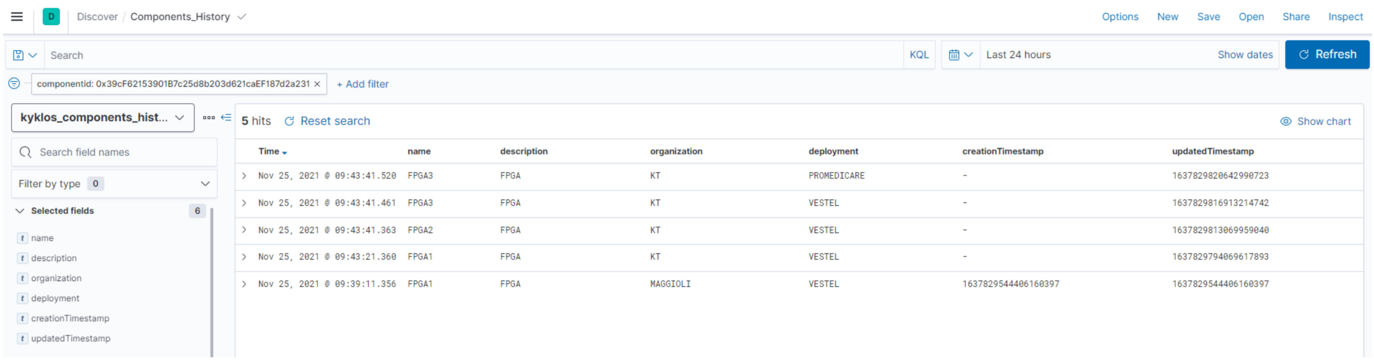
@timestamp per day	id	Name	Organization	Description	Deployment	Owner	creationTime	updatedTime
2021-11-25	0x1De230Ea5cA9A476fF5...	Sensor2	TECNALIA	Humidity	GRC	0x065bD8E7E026AC1f6cE...	1637828213877281276	163782821387728...
2021-11-25	0x376ff80089B29133A7bf...	Sensor	TECNALIA	Temperature	GRC	0x10592b969Eb34CD0179...	1637828096780637377	1637828096780637377
2021-11-25	0x39cF62153901B7c25d8...	FPGA1	MAGGIOLI	FPGA	VESTEL	0x6A3470B01245DE7F104...	1637829544406160397	1637829544406160397
2021-11-22	0x0dE4b77C6683Bd29fAa...	MS	SIMAVI	Example	VESTEL	0xE3BE706286E1f8F9D85...	1637567298746205558	1637567354674844752

Registered Features

@timestamp per day	id	Feature Name	Feature Value	updatedTime
2021-11-22	0x5f5A2dBF4BdAbDE3f05a0D67c5910B7b87c0DFE3	os	Linux	1637571359003914091
2021-11-22	0x5f5A2dBF4BdAbDE3f05a0D67c5910B7b87c0DFE3	temp	100	1637571328249336166
2021-11-22	0x9f5c8103043C2D02eaC9325d36556B32d53B6e66	os	Windows	1637573028407700336

Figure 4. Example of a graphical monitor client (general dashboard from KYKLOS 4.0 project).

- The history report, where all updates associated to an audit trail since its creation can be consulted, being able to detect all changes occurred and when they took place. It is not necessary to add details about who has updated the component information because only the owner of the audit trail can update its associated audit trail. The history report on its audit trail is accessed from the general dashboard by accessing each specific audit trail ID (shown as blue links in Figure 4). Figure 5 shows an example of the history for a component registered in the KYKLOS 4.0 project.



componentid: 0x39cF62153901B7c25d8b203d621caEF187d2a231

5 hits

Time	name	description	organization	deployment	creationTimestamp	updatedTimestamp
Nov 25, 2021 @ 09:43:41.520	FPGA3	FPGA	KT	PROMEDICARE	-	1637829820642990723
Nov 25, 2021 @ 09:43:41.461	FPGA3	FPGA	KT	VESTEL	-	1637829816913214742
Nov 25, 2021 @ 09:43:41.363	FPGA2	FPGA	KT	VESTEL	-	1637829813069959040
Nov 25, 2021 @ 09:43:21.360	FPGA1	FPGA	KT	VESTEL	-	1637829794069617893
Nov 25, 2021 @ 09:39:11.356	FPGA1	FPGA	MAGGIOLI	VESTEL	1637829544406160397	1637829544406160397

Figure 5. Example of a graphical monitor client (history report from KYKLOS 4.0 project).

5. Audit Trail Mechanism Operation

As explained in Section 4, there are three types of actors interacting with the audit trail mechanism:

- Administrators: people, automated systems, etc. who manage the system and authorize users to use it.
- Authorized users: people, automated systems, etc. who provide data to be registered in the Blockchain-based auditing mechanism and consume it as required. Authorized users always need the Blockchain client to interact with the Blockchain.
- External data consumers: people who consume the data registered in the Blockchain-based audit trail mechanism. They do not need any Blockchain client as they do not access the Blockchain directly, but they access the Blockchain monitor, which has a copy of all-important events occurring on the Blockchain by means of the subscribed Blockchain-based events. They cannot register or modify audit trails on the Blockchain. They only need to access the Blockchain monitor client by means of a web browser.

Figure 6 shows the sequence diagram of the audit trail mechanism operation in which the most important events are depicted. All events shown in Figure 6 are automatic except for the interactions with the three actors mentioned above.

The operation process has the following steps:

- An administrator must authorize a user to use the audit trail mechanism.
- The authorized user can register a new audit trail (new data will be registered in the Blockchain)
- The authorized user can update an existing audit trail (data recorded in the Blockchain will be updated).
- The authorized user can obtain the current version of an existing audit trail.
- An external data consumer can obtain the information recorded on the Blockchain (main dashboard or history report) by means of the web-based Blockchain monitor client (authentication is required).

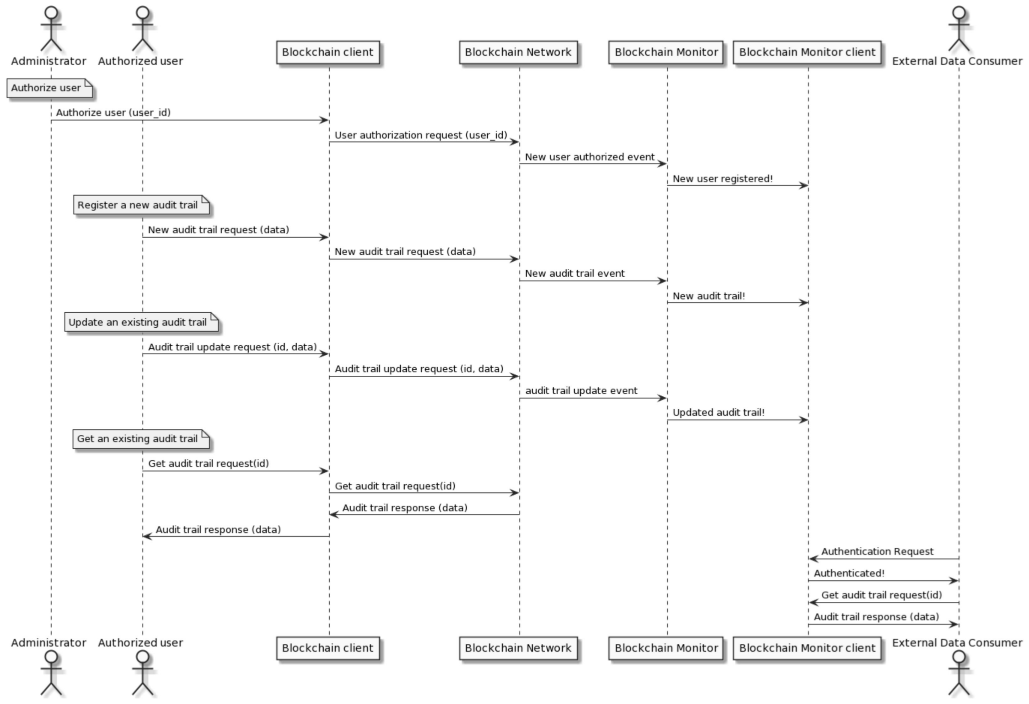
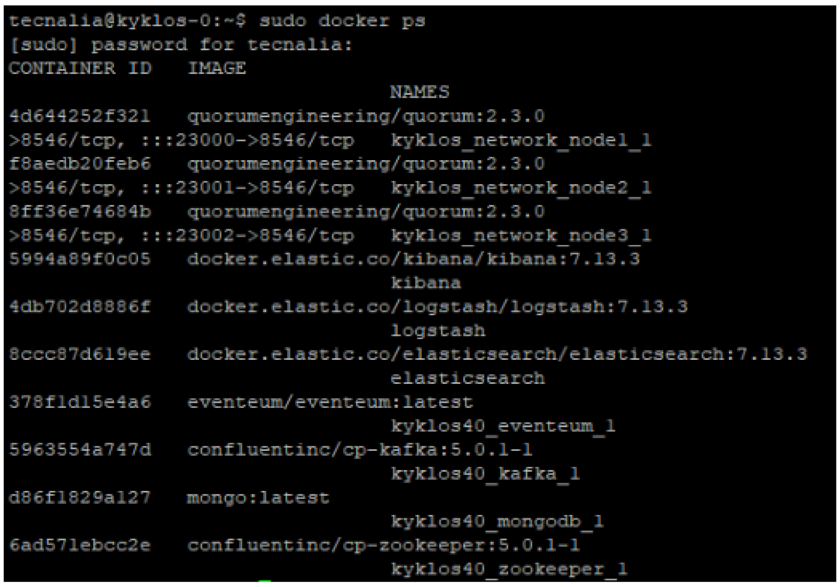

Administrator
Authorized user
Blockchain client
Blockchain Network
Blockchain Monitor
Blockchain Monitor client
External Data Consumer
Authorize user
Authorize user (user_id)
User authorization request (user_id)
New user authorized event
New user registered!
Register a new audit trail
New audit trail request (data)
New audit trail request (data)
New audit trail event
New audit trail!
Update an existing audit trail
Audit trail update request (id, data)
Audit trail update request (id, data)
audit trail update event
Updated audit trail!
Get an existing audit trail
Get audit trail request(id)
Get audit trail request(id)
Audit trail response (data)
Audit trail response (data)
Authentication Request
Authenticated!
Get audit trail request(id)
Audit trail response (data)

Figure 6. Audit trail mechanism operation sequence diagram.

6. Validation Experiment

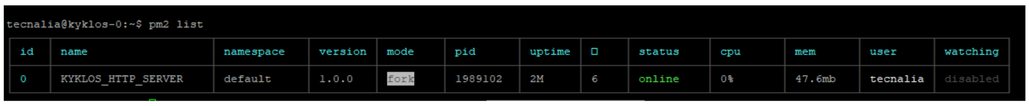
The correct operation of the proposed audit trail mechanism was validated through the level of compliance with the requirements defined in Section 2.3 for a Blockchain-based audit system that enhances the security and usability of current audit solutions. For this purpose, a controlled laboratory validation scenario was deployed as follows:

- Blockchain network: Three Quorum Blockchain nodes were locally deployed as docker images for validation purposes (kyklos_network_node1_1, kyklos_network_node2_1 and kyklos_network_node3_1 in Figure 7); however, in real deployments, additional nodes managed by different owners are recommended to improve decentralization and governance of the Blockchain network. The implemented smart contracts with the functionalities described in Section 4 were registered (deployed) on the Blockchain network.
- Blockchain monitor and Blockchain monitor client: One instance of each of the required services for the Blockchain monitor and Blockchain monitor client were locally deployed as docker images for validation purposes; in real deployments, at least the Blockchain monitor client must be publicly accessible.
- Eventeum was deployed through the kyklos40_eventeum_1 docker image in Figure 7. The internal operation of Eventeum requires Mongo DB [51] (kyklos40_mongodb_1 in Figure 7).
- Kafka was deployed through the kyklos40_kafka_1 docker image in Figure 7. The internal operation of Kafka requires Zookeeper [52] (kyklos40_zookeeper_1 in Figure 7).
- Logstash, Elasticsearch, and Kibana, as part of the Elastic Stack, were deployed through the Logstash, Elasticsearch, and Kibana docker images in Figure 7, respectively.
- Blockchain client: One Blockchain client instance was locally deployed as a Node.js application for validation purposes by means of the PM2 process manager [53] (KYKLOS_HTTP_SERVER in Figure 8); however, in real deployment, any component that is likely to provide audit information should deploy its own instance of Blockchain client.



```
tecnalia@kyklos-0:~$ sudo docker ps
[sudo] password for tecnalia:
CONTAINER ID   IMAGE
                                 NAMES
4d644252f321   quorumengineering/quorum:2.3.0
>8546/tcp, :::23000->8546/tcp   kyklos_network_node1_1
f8aedb20feb6   quorumengineering/quorum:2.3.0
>8546/tcp, :::23001->8546/tcp   kyklos_network_node2_1
8ff36e74684b   quorumengineering/quorum:2.3.0
>8546/tcp, :::23002->8546/tcp   kyklos_network_node3_1
5994a89f0c05   docker.elastic.co/kibana/kibana:7.13.3
                                 kibana
4db702d8886f   docker.elastic.co/logstash/logstash:7.13.3
                                 logstash
8ccc87d619ee   docker.elastic.co/elasticsearch/elasticsearch:7.13.3
                                 elasticsearch
378f1d15e4a6   eventeum/eventeum:latest
                                 kyklos40_eventeum_1
5963554a747d   confluentinc/cp-kafka:5.0.1-1
                                 kyklos40_kafka_1
d86f1829a127   mongo:latest
                                 kyklos40_mongodb_1
6ad571ebcc2e   confluentinc/cp-zookeeper:5.0.1-1
                                 kyklos40_zookeeper_1
```

Figure 7. Docker images deployed in the validation experiment.



```
tecnalia@kyklos-0:~$ pm2 list
│ id │ name               │ namespace │ version │ mode │ pid     │ uptime │ ↺ │ status │ cpu │ mem    │ user     │ watching │
│ 0  │ KYKLOS_HTTP_SERVER │ default   │ 1.0.0   │ fork │ 1989102 │ 2M     │ 6 │ online │ 0%  │ 47.6mb │ tecnalia │ disabled │
```

Figure 8. Node.js application from the validation experiment.

Table 3 summarizes how the requirements identified in Table 2 (Section 2.3) were addressed in the validation experiment, resulting in a secure and user-friendly general-purpose Blockchain-based auditing system.

Table 3. Blockchain-based audit trail mechanism requirements coverage.

Requirement Name	How It Was Addressed
Inalterability (integrity)	Blockchain owns inherent characteristic of integrity (the information recorded on the Blockchain cannot be modified/altered at all).
Long-term maintenance	Blockchain owns the inherent characteristic of long-term maintenance (the information recorded on the Blockchain cannot be erased).
Assurance of origin	Blockchain has the inherent feature of guarantee of origin (the information recorded on the Blockchain is always associated with a specific account on the Blockchain).
Simultaneity	Each stakeholder providing or consuming information has its own Blockchain client instance, while the Blockchain transaction supports simultaneous write and read operations.
Availability	Blockchain networks are distributed by design; thus, high availability is guaranteed.
Confidentiality	SSL is used in all communications.
Access control	The auditing system is designed in such a way that only the data source can update or retrieve its own data through the DApp, while access through the Blockchain monitor client is authenticated; hence, a proper account is required to access the graphical information. Thus, data access control is guaranteed.
Usability	Graphical information (tables) is provided in the Blockchain monitor client.
Multidomain	The Blockchain monitor allows the recording of any kind of information thanks to the name-value pair in the data model (multisector, multidomain).

Table 3. *Cont.*

Requirement Name	How It Was Addressed
Abstraction of the Blockchain	A Blockchain monitor is included in the audit trail system to allow users without any Blockchain account to access the Blockchain-based audit trail data.
REST API	An REST API is implemented in the Blockchain client to facilitate the deployment, integration, and use of the auditing system by any stakeholder.

As a result, the proposed Blockchain-based auditing system solves the main limitations of the current Blockchain-based auditing solutions that were identified in the state-of-the-art analysis (Section 2.2). Table 4 gathers how the proposed auditing mechanism addresses the identified limitations.

Table 4. Improvements over the state-of the-art Blockchain-based audit trail approaches.

Study	Main Limitation	Improvement
[31]	Lack of technical aspects specific to the Blockchain technology.	Detailed technical description is provided.
[32]	Limited to banking context (broader applicability limited). Usability is not considered.	The proposed auditing system is multidomain and multisector. Usability is provided by means of the Blockchain monitor and Blockchain monitor client.
[33–36]	Availability is at risk (traditional databases). Two systems are required; complexity increases (synchronization is required).	The auditing functionality and the registered data are only recorded on a distributed Blockchain; thus, high availability is guaranteed, and complexity is limited to the Blockchain technology.
[37]	Limited application. Usability is not considered.	The proposed auditing system is multidomain and multisector. Usability is provided by means of the Blockchain monitor and Blockchain monitor client.
[38]	Auditing functionality is not secure enough (Blockchain only saves data but does not execute auditing functions; an additional log server is needed). Usability is not considered.	The auditing functionality is provided through smart contracts directly registered on the Blockchain; thus, it is secure and tamper-resistant. Usability is provided by means of the Blockchain monitor and Blockchain monitor client.

7. Conclusions

This paper improves the security and usability of current audit trail solutions by designing and implementing a novel general-purpose Blockchain-based audit trail mechanism. On the one hand, the proposed mechanism leverages Blockchain inherent security features such as integrity, traceability, availability, and nonrepudiation to ensure a high security level of audit trails. On the other hand, a Blockchain monitor is included to enable high usability and isolate users from the use of Blockchain as backbone. As a result, a real prototype of a general-purpose Blockchain-based audit trail mechanism that contributes to more reliable, secure, and user-friendly audit trails was presented and described in detail with emphasis on the technical particularities of using the Blockchain technology. Additionally, their improvements over the state-of-the-art methods were identified.

The proposed auditing mechanism will be validated within the Horizon 2020 project KYKLOS 4.0 (Grant agreement ID: 872570) to provide audit trails of technical manufacturing components configurations. However, it is not a domain-specific system and can be used in any other type of ecosystem, such as banking, healthcare, the food industry, etc. Further research is needed in terms of performance evaluation (latency and gas consumption) in production environments.

The research will continue by improving the proposed mechanism by defining security detection rules, which would lead to a new approach to security information and event management (SIEM) systems where data monitoring and traceability are essential.

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

Funding: This research was funded by the European Commission, grant number 872570 (H2020 KYKLOS 4.0 project).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

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

References

1. Costin, A.; Zaddach, J. Iot Malware: Comprehensive Survey, Analysis Framework and Case Studies. Available online: https://i.blackhat.com/us-18/Thu-August-9/us-18-Costin-Zaddach-IoT-Malware-Comprehensive-Survey-Analysis-Framework-and-Case-Studies-wp.pdf (accessed on 21 October 2021).
2. Zhang, Z.K.; Cho, M.C.Y.; Wang, C.W.; Hsu, C.W.; Chen, C.K.; Shieh, S. IoT security: Ongoing challenges and research opportunities. In Proceedings of the 2014 IEEE 7th International Conference on Service-Oriented Computing and Applications, Matsue, Japan, 17–19 November 2014.
3. Ahmad, R.; Alsmadi, I. Machine learning approaches to IoT security: A systematic literature review. *IEEE Internet Things* **2021**, *14*, 100365.
4. Valaboju, Y. A Comprehensive Study on Iot Architectures and Iot Security. *Parishodh J.* **2019**, *VIII*, 57–66.
5. Mohanta, B.K.; Jena, D.; Satapathy, U.; Patnaik, S. Survey on IoT security: Challenges and solution using machine learn-ing, artificial intelligence and blockchain technology. *IEEE Internet Things* **2020**, *11*, 100227.
6. Nadir, I.; Ahmad, Z.; Mahmood, H.; Shah, G.A.; Shahzad, F.; Umair, M.; Gulzar, U. An Auditing Framework for Vul-nerability Analysis of IoT System. In Proceedings of the 2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), Stockholm, Sweden, 17–19 June 2019.
7. Wolf, Z.R. Exploring the audit trail for qualitative investigations. *Nurse Educ.* **2003**, *28*, 175–178. [CrossRef] [PubMed]
8. Kennedy, G.E.; Judd, T.S. Making sense of audit trail data. *Australas. J. Educ. Technol.* **2004**, *20*, 18–32. [CrossRef]
9. Duncan, R.A.K.; Whittington, M. Enhancing cloud security and privacy: The power and the weakness of the audit trail. Available online: https://aura.abdn.ac.uk/bitstream/handle/2164/8061/cloud_computing_2016_6_20_20063.pdf?sequence=1 (accessed on 21 October 2021).
10. Khanuja, H.K.; Adane, D.S. S. Database security threats and challenges in database forensic: A survey. In Proceedings of 2011 International Conference on Advancements in Information Technology (AIT 2011), Chennai, India, 17–18 December 2011.
11. Nikolaou, C.N.; Marazakis, M.; Georgiannakis, G. Transaction routing for distributed OLTP systems: Survey and recent results. *Inf. Sci.* **1997**, *97*, 45–82. [CrossRef]
12. Lee, K.H.; Zhang, X.; Xu, D. LogGC: Garbage collecting audit log. In Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, Berlin, Germany, 4–8 November 2013; pp. 1005–1016.
13. Zheng, Z.; Xie, S.; Dai, H.N.; Chen, X.; Wang, H. Blockchain challenges and opportunities: A survey. *Int. J. Web Grid Serv.* **2018**, *14*, 352–375. [CrossRef]
14. Ljunggren, N. Improving the Usability of Secure Information Storing within Blockchain Applications. Master's Thesis, Lund University, Lund, Sweden, 2019.
15. Clincy, V.; Shahriar, H. Blockchain development platform comparison. In Proceedings of the 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), Milwaukee, WI, USA, 15–19 July 2019; IEEE: Piscataway, NJ, USA, 2019; Volume 1, pp. 922–923.
16. 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]
17. Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 21 October 2021).
18. Wood, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Proj. Yellow Pap.* **2014**, *151*, 1–32.
19. Cong, L.W.; He, Z. Blockchain disruption and smart contracts. *Rev. Financ. Stud.* **2019**, *32*, 1754–1797. [CrossRef]

20. 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.
21. Morgan, J.P. Quorum Whitepaper. Technical Report. 2016. Available online: https://github.com/jpmorganchase/quorum-docs/blob/master/Quorum%20Whitepaper%20v0.1.pdf (accessed on 11 November 2021).
22. Consensys Quorum. Available online: https://consensys.net/quorum/ (accessed on 11 November 2021).
23. Lombardi, R.; de Villiers, C.; Moscariello, N.; Pizzo, M. The disruption of blockchain in auditing—A systematic literature review and an agenda for future research. Available online: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3834838 (accessed on 21 October 2021).
24. Tiron-Tudor, A.; Deliu, D.; Farcane, N.; Dontu, A. Managing change with and through blockchain in accountancy organizations: A systematic literature review. Available online: https://www.emerald.com/insight/content/doi/10.1108/JOCM-10-2020-0302/full/html (accessed on 21 October 2021).
25. Pimentel, E.; Boulianne, E.; Eskandari, S.; Clark, J. Systemizing the challenges of auditing blockchain-based assets. *J. Inf. Syst.* **2021**, *35*, 61–75. [CrossRef]
26. Psaila, S. Blockchain: A Game Changer for Audit Processes. Available online: https://www2.deloitte.com/mt/en/pages/audit/articles/mt-blockchain-a-game-changer-for-audit.html (accessed on 11 November 2021).
27. Chartered Professional Accountants of Canada (CPA Canada) and the American Institute of CPAs (AICPA). Available online: https://us.aicpa.org/content/dam/aicpa/interestareas/frc/assuranceadvisoryservices/downloadabledocuments/blockchain-technology-and-its-potential-impact-on-the-audit-and-assurance-profession.pdf (accessed on 11 November 2021).
28. Dai, J. Three Essays on Audit Technology: Audit 4.0, Blockchain, and Audit App. Ph.D. Thesis, Rutgers University-Graduate School-Newark, New Brunswick, NJ, USA, 2017.
29. Fanning, K.; Centers, D.P. Blockchain and Its Coming Impact on Financial Services. *J. Corp. Account. Financ.* **2016**, *27*, 53–57. [CrossRef]
30. Kiviat, T.I. Beyond Bitcoin: Issues in Regulating Blockchain Transactions. *Duke Law J.* **2015**, *65*, 569.
31. Ahmad, A.; Saad, M.; Bassiouni, M.; Mohaisen, A. Towards blockchain-driven, secure and transparent audit logs. In Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, New York, NY, USA, 5–7 November 2018; pp. 443–448.
32. Peters, G.W.; Panayi, E. Understanding Modern Banking Ledgers through Blockchain Technologies: Future of Transaction Processing and Smart Contracts on the Internet of Money. In *Banking Beyond Banks and Money*; Working paper; Cornell University: Ithaca, NY, USA, 2015.
33. Sutton, A.; Samavi, R. Blockchain Enabled Privacy Audit Logs. In Proceedings of the International Semantic Web Conference ISWC, Vienna, Austria, 21–25 October 2017; pp. 645–660. [CrossRef]
34. VishwaVidyapeetham, A. A blockchain and ipfs based framework for secure research record keeping. *Int. J. Pure Appl. Math.* **2018**, *119*, 1437–1442.
35. Li, J.; Wu, J.; Jiang, G.; Srikanthan, T. Blockchain-based public auditing for big data in cloud storage. *Inf. Process. Manag.* **2020**, *57*, 102382.
36. Cucurull, J.; Puiggalí, J. Distributed Immutabilization of Secure Logs. In *Security and Trust Management. STM 2016*; Lecture Notes in Computer, Science; Barthe, G., Markatos, E., Samarati, P., Eds.; Springer: Cham, Switzerland, 2016; Volume 9871. [CrossRef]
37. Wang, Y.; Li, J.; Yan, Y.; Chen, X.; Yu, F.; Zhao, S.; Yu, T.; Feng, K. A semi-centralized blockchain system with multi-chain for auditing communications of Wide Area Protection System. *PLoS ONE* **2021**, *16*, e0245560. [CrossRef] [PubMed]
38. Ali, A.; Khan, A.; Ahmed, M.; Jeon, G. BCALS: Blockchain-based secure log management system for cloud computing. Available online: https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.4272 (accessed on 21 October 2021).
39. Dannen, C. *Introducing Ethereum and Solidity*; Apress: Berkeley, CA, USA, 2017; Volume 318, Available online: https://link.springer.com/content/pdf/bfm%253A978-1-4842-2535-6%252F1.pdf (accessed on 21 October 2021).
40. Van Leeuwen, J.; Wiedermann, J. The Turing machine paradigm in contemporary computing. In *Mathematics Unlimited—2001 and Beyond*; Springer: Berlin/Heidelberg, Germany, 2001; pp. 1139–1155.
41. Raft Consensus Protocol. Available online: https://docs.goquorum.consensys.net/en/stable/Concepts/Consensus/Raft/ (accessed on 11 November 2021).
42. Istanbul BFT. Available online: https://github.com/ethereum/EIPs/issues/650 (accessed on 11 November 2021).
43. Kiayias, A.; Zindros, D. Proof-of-work sidechains. In Proceedings of the International Conference on Financial Cryptography and Data Security, Frigate Bay, Saint Kitts and Nevis, 18–22 February 2019; pp. 21–34.
44. Baliga, A.; Subhod, I.; Kamat, P.; Chatterjee, S. Performance evaluation of the quorum blockchain platform. *arXiv* **2018**, arXiv:1809.03421.
45. web3.js—Ethereum JavaScript API. Available online: https://web3js.readthedocs.io/en/v1.4.0/ (accessed on 11 November 2021).
46. Eventeum Source Code. Available online: https://github.com/eventeum/eventeum (accessed on 11 November 2021).
47. Apache Kafka. Available online: https://kafka.apache.org/ (accessed on 11 November 2021).
48. Logstash. Available online: https://www.elastic.co/es/logstash/ (accessed on 11 November 2021).
49. Elasticsearch. Available online: https://www.elastic.co/es/what-is/elasticsearch (accessed on 11 November 2021).
50. Kibana. Available online: https://www.elastic.co/es/kibana/ (accessed on 11 November 2021).

51. Mongo DB. Available online: https://www.mongodb.com/ (accessed on 11 November 2021).
52. Zookeeper. Available online: https://zookeeper.apache.org/ (accessed on 11 November 2021).
53. PM2: Advanced, Production Process Manager for Node.js. Available online: https://pm2.keymetrics.io/ (accessed on 11 November 2021).