A Proof of Authority Blockchain Protocol for Secure Logging

Gabriel Loyola Daiqui
Seminar Inverse Transparency (WS 19/20)
Advisor: Valentin Zieglmeier
gabriel.loyola-daiqui@tum.de

Arnau Oller Prat
Seminar Inverse Transparency (WS 19/20)
Advisor: Valentin Zieglmeier
arnau.oller01@estudiant.upf.edu

***Abstract*—Due to the continuous interaction with internet services a person produces a high quantity of data every day. Some of this data is of a more private nature, nonetheless, we do not know who has access to it. In some cases, we do not even know of the existence of such sensitive data. Due to the current lack of transparency on the usage of online data and new regulations we propose a solution making use of Hyperledger Fabric, a permissioned blockchain with a Proof-of-Authority consensus algorithm, to create this new form of transparency for data logging while maintaining a secure and scaleable network. Such a network can be used internally at a company to abide by the new rules of the GDRP while facilitating the fair access to an employee's data for both the employer and employee.**

***Index Terms*—Hyperledger Fabric, blockchain, privacy, logging**

I. Introduction

Blockchain is a great example of disruptive technology. It first debuted in Bitcoin [1], a popular decentralized cryptocurrency. Today it can be found in many different industries ranging from financial companies [2], to video platforms [3] and even critical infrastructure security.

An essential aspect of information security is secure data logging, which should abide in privacy regulations like the GDPR [4] while at the same time gathering as much information as possible. Data logging should provide traceability of who created, read, modified or deleted a file, while at the same time, respecting data privacy.

A good logging solution has to fulfill the following requirements: it has to (i) be able to handle a partial system failure, (ii) guarantee data consistency, (iii) guarantee tampering protection, and (iv) protect the log access. The permissioned blockchain protocol meets the above conditions and is therefore predesignated to be used for data logging.

This paper will discuss the different aspects to be considered in the design and implementation of a secure logging protocol based on a permissioned blockchain using a Proof of Authority consensus algorithms.

II. Use Case

In the context of a company, an employer receives a significant amount of private data when a new employee joins the company. Different departments need to access parts of that data to be able to perform their work. For example, human resources need the social security number to verify that the employee has a valid work permit, while payroll requires the bank account information and tax identification number to be able to pay the salary to the employee.

With the current systems, it is therefore nearly impossible to track which personal and/or confidential information has been accessed and who has accessed at any point in time. This makes it difficult to verify that the personal data was handled in accordance with the applicable laws and regulations.

A possible solution to this problem is to use a logging system that provides oversight of the data access. Such a system would log what information is accessed by which person or system and why the information was requested. These data logs offer multiple benefits:

- Transparency: Inform the employee when, why and by whom their personal data was requested
- Auditing: Provides the possibility to keep close track to and early detection of any defective/malicious system that tries to access confidential or strictly confidential data without a valid reason and ensures that the responsible entities are held accountable and proper protection is re-established.
- Privacy incident control: If personal data is leaked, the logs provide a list of all possible suspects. Privacy incident control additionally provides the means of early detection and correction.
- Data analytics: The information recorded on the logs can be used, for example, to gain a better understanding of the relations among employees

Our goal is to create a logging solution that provides transparency to the employee, and gives them the possibility of verifying the correct usage of their personal data.

III. Problem Statement

In a company environment, an employee trusts the company with sensitive data, such as their address, age, marriage status, education, social security number, and bank account information.

Currently, the employee has to hope that the data is being kept confidential and secure since there is no way for them to check who accessed it. If this problem is not addressed properly, a malicious or careless employer could edit, publish or even sell private information and the company would not be able to prove it to hold them accountable.

To address this issue, we propose a blockchain logging approach using the Hyperledger Fabric framework [5]. This approach uses a network of peers, organizations, endorsers and a channel to allow for the modification and retrieval of data from the ledger [6]. The application should also prevent any unauthorized personnel from accessing the sensitive data, by only allowing the employer and employee to view or edit their data. Additionally, the system should keep track of all changes, modifications and queries that have occurred on the ledger.

IV. Why Blockchains?

Blockchains have unique advantages that make them predestined for usage in data logging. These advantages include:

A. Fault tolerance

Both Proof of Work (PoW) and Proof of Authority (PoA) blockchains use consensus algorithms that are part of the Byzantine Fault Tolerant (BFT) family [7]. This means that the network can still provide accurate and reliable results even if a certain number of nodes are not working as intended [8], [9], irrespective of the reason for the failure, be it a software bug or a malicious node. Furthermore, blockchains can be run in a distributed peer-to-peer network, minimizing the risk of data loss in the event of a system failure.

B. Data consistency

Blockchains are considered consistent as long as there are no forks [10]. A fork occurs if the peers disagree on the block order or block content resulting in the blockchain being heterogeneous. Resolving a fork means selecting the 'correct' branch and discarding the other. As a consequence, all transactions stored in the discarded branch are no longer valid. Each consensus algorithm has a different approach to minimize the risk of forking and handling situations where forks have occurred [7].

C. Tamper protection

Blockchains guarantee tamper protection by including the hash of the previous block in the current one [8]. Any change to block n results in a different hash for n, which means that the hash of n stored in the block $n+1$ does not match with the altered block n. This mechanism allows the network to easily recognize tampering and reject the fraudulent block, as long as it is not the latest block.

While it is theoretically possible to find a colliding hash, in the practice this is computationally expensive. For example, finding a collision in the outdated SHA-1 hashing algorithm took 1 year and 110 GPUs [11]. This security risk can be minimized by using a trusted hashing algorithm like SHA-256.

D. Privacy

Privacy can be ensured by using a permissioned blockchain [9], where only a restricted set of nodes are granted read or write access to the blockchain, protecting the sensitive data in logs.

Permissioned blockchains are usually implemented by adding a central authority that governs the actions performed by trusted agents. In contrast to permissionless blockchains, the permissioned blockchains are controlled by a central authority that restrict the participants, the allowed operations, and selects verified nodes that are allowed to create or validate transactions. The main advantages of permissioned blockchains are the central governance structure and the increased performance due to the limited number of allowed nodes.

V. Different Blockchain Implementations

There are a few consensus types for blockchains. The most know, since it's the consensus used in Bitcoin, is Proof of Work (PoW), in which to add transactions to the blockchain a computation is needed. Probably the second most know is Proof of Stake, in which the ones that secure the blockchain network need "stakes" that are usually cryptocurrencies. Another solution is Proof of Authority, where just some selected nodes have the power to manage the network.

While we found more blockchain solutions using PoW instead of PoA, PoW is not appropriate for logging data for multiple reasons.

a) Energy and computational waste: PoW relies on brute force for finding a hash with enough leading zeros as their Proof of Work [1], which leads to high energy and computational costs [12]

b) Reduced transactions per second: PoW results in reduced transactions per second due to its nature of relying on heavy computation to find a consensus [9].

The PoA algorithms solve these issues; in contrast to PoW that allows every miner to add blocks to the blockchain, PoA only allows a trusted set of so-called authorities to propose and commit blocks [9]. The PoA proposal process hence consumes less energy and computational resources since the consensus is reached by exchanging messages. The lack of heavy computational costs leads to a higher transactional throughput as well.

While PoS also solve some of the PoW issues, PoA implementations can be affected by stake-bleeding, where a single node could control the entire network [13], making them not suitable for logging.

VI. Existing PoA Blockchain Solutions

The boom of Bitcoin lead to an abundance of products utilizing blockchains, some more useful than others. We picked three PoA solutions that looked the most promising to us:

A. Parity

Parity is an Ethereum Client written in Rust [14] and based on the Aura (Authority Round) [15] consensus algorithm. Its main features are:

1) Modularity: Ethereum is being developed independent of any specific field of application and therefore is a generic platform for all kinds of transactions, including logging. It is only modular to an extent though, forcing any application to handle transaction costs, know as fuel costs in Ethereum, even when the application does not need it.

2) High performance: Since Aura is a PoA algorithm, it uses less energy and computational resources than a traditional PoW solution. It is thus faster and more efficient [7].

3) Open-source: Parity is an open-source project [16] with all corresponding benefits. Additionally, the core code is reviewed by an auditing firm called Trail of Bits [17].

4) Smart contracts: Because Parity is based on Ethereum, it allows for the usage of the specially created smart contract programming languages Solidity and Vyper [18].

Disadvantages: Parity inherits some drawbacks from Ethereum, like fuel cost which is not needed in a logging solution. Furthermore, Ethereum does not support permissioned blockchains.

Known issues: Synchronicity is crucial because the time step leaders are chosen based on UNIX time. A time drift between the authorities causes phases in which the nodes are not in agreement on who the current leader is. In the best-case scenario, the faulty nodes are in the minority and can be voted out by the honest majority. In the worst case, the amount of faulty nodes is equal to the number of correct ones or the majority contains enough Byzantine nodes(nodes misbehaving) blocking the removal process. This causes the inability to vote out any authority, which leads to a fork causing data inconsistency [7].

Another weakness is the cloning attack [19] where a Byzantine node partitions the network into two disjoint sets and clones itself into both partitions. The separated nodes continue to work independently, possibly adding different blocks to the chain. Once the partition is revoked, one fork is discarded, leading to the loss of valid blocks.

B. Geth

Geth is an Ethereum client written in Golang, using the Clique consensus algorithm [20]. The main features are:

1) Modularity: Since Geth is based on Ethereum as well, it is just as modular as Parity.

2) High performance: Clique is even faster than Aura because fewer messages need to be exchanged between nodes before a consensus is reached [7].

3) Open source: Geth is an open-source project [21] as well, meaning that it's possible to review the entire source code, making it more secure and trustworthy.

Disadvantages: Geth is based on Ethereum as well, which means that it has the same drawbacks as Parity and no support for permissioned blockchains.

Know issues: Since forks are possible in Clique, the PoA blockchain is only eventually consistent in contrast to the PoW blockchain which is always consistent. Additionally Clique is, like Aura, affected by the cloning attack problem [19].

C. Hyperledger Fabric

Hyperledger Fabric (HLF) [5] is a modular blockchain framework and is one of the blockchain solutions developed by the Linux Foundation [22]. It's advantages include:

1) Modularity: HLF is a modular platform that allows components (like the consensus or membership service) to be easily integrated with a plug and play support. This focus on modularity leads to a high configurability and possibly improved performance over standardized solutions.

2) Privacy: HLF is a permissioned blockchain, meaning that only a selected set of nodes are allowed to read or write on the blockchain thus preventing unauthorized access to the stored data.

3) Accountability: Since HLF uses a permissioned blockchain, every node that interacts with the network is known to the central authority. Hence a misbehaving peer can be easily identified and corrected.

4) Design: Unlike Geth, Parity or any other Ethereum client, HLF is specifically designed for enterprise use meaning that unnecessary burden like transaction fees do not exist, allowing for a more streamlined solution.

5) Channels: HLF features the creation of channels between the nodes, bypassing a central authority, and thus allowing for the private sharing of sensitive data. Furthermore, it allows sharing only certain data with other nodes.

6) High performance: HLF is a high-performance blockchain and can handle ten thousand transactions a second with peers being distributed on different continents [23].

7) Open-source: HLF is an open-source project with all the benefits it implies(free to use, collaboration, code availability) and also with the extra layer of trust it gives to use one of the blockchain solutions supported by the Linux Foundation projects [22].

8) Assets: HLF allows to represent any kind of asset in it, it can be used to represent tangible assets such as minerals [24] and food [25] and also for intangible asssets such as logging credentials [26].

9) Chaincode: The program that usually handles business logic of the blockchain, it allows us to read and write to the ledger, and can be written in Go, Java or NodeJs. With it's most common application, business logic, it is really similar to a smart contract [27], like in Ethereum.

D. Selected PoA blockchain paradigm

Geth and Aura are not suitable for data logging due to the risk of data loss. Furthermore, they both do not offer native support for permissioned blockchains. In order to ensure data privacy, a significant amount of modifications would be needed. Hyperledger Fabric has the same advantages as Geth and Aura, while not being affected by time drift or data loss. Therefore we choose Hyperledger Fabric as the blockchain paradigm to use for secure logging.

VII. Proof of Concept Implementation

We implemented a proof of concept, demonstrating that it's possible to build a secure and transparent logging system utilizing Hyperledger Fabric that fulfills the requirements of the use case explained in section II.

A. *Structure of the implementation*

1) Network: The network is composed of two different entities that are connected using PrivacyNet, a Hyperledger Fabric blockchain platform. An overview can be found in Fig.1.

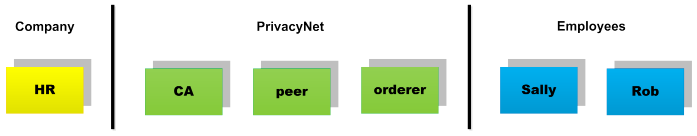

Fig. 1. A simplified overview of the proposed blockchain network

The network has two entities. One entity is responsible for the company, which is here represented by the human resource department.

The other entity is responsible for all the employees, each employee represents themselves.

The HLF blockchain consists of a central authority, different peers and an orderer.

The central authority (CA) has the sole permission to create new users for both, the company and the employees. Due to HLF's modularity, the CA can be integrated into existing solutions.

The orderer's function is to receive all data that is to be written to the blockchain and order the transaction. Since the consensus algorithm used in HLF is deterministic, a validated block from the orderer is final and correct.

2) Peers: The peers receive the blocks from the orderer and add them to their ledger, which contains all transactions since the start of the network. Furthermore, each peer also contains the chaincode, allowing the peer to run required smart contracts on which the application is based. Executing every action in the given order stored in the blockchain generates the world state. Due to HLF's deterministic consensus algorithm, the world state is the same across all peers. Figure 2 shows the different peer components.

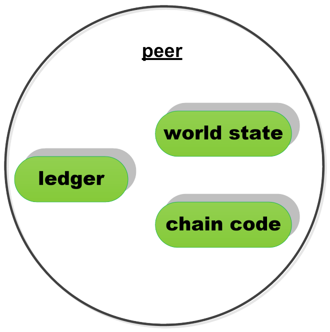

Fig. 2. A simplified view of a peer

B. *Company use case*

Let's imagine there is a company called 'TechCo', which is looking for a way to comply with the GDPR into their internal human resource system. They use Hyperledger Fabric to write an application that allows for better data and access control while being as transparent as possible. In this example, we will refer to the employee John and the company is represented by Thomas, a member of the HR department.

1) How the company uses the application: Thomas wants to check if John is reaching retirement age, to know if they should begin looking for a replacement. In order to do this, the following will take place:

- Thomas logs into the system using the credentials that were created for him by the central authority when Thomas moved to the corresponding role. Since he is an authorized employee and has been assigned the corresponding role in the company, he is allowed to interact with the network.
- When he requests to see John's contract, he needs to enter a justification for viewing it
- The system checks, if Thomas is allowed to view John's contract. Thomas is a registered member of the HR department, allowing him to see the contract.
- Thomas checks John's retirement status, while the application saves Thomas access to John's contract into the blockchain.

2) How an employee uses the application: John is a privacy-oriented individual and wants to put his mind at ease by checking the access to his personal contract. The process is as follows:

- John logs into the application by providing his certificate
- He sends a request to view his contract
- The application checks if he is allowed to see the requested contract. Access is granted because it is his own contract.
- John receives a list of all access requests his contract, together with the justification provided with each request. In this example he will see that Thomas requested to view his contract in order to know if John will retire soon.

C. *Implementation decisions*

Based on the above-described problem statement, the following implementation decisions were made:

1) Hyperledger version: As of December 2019 the new Hyperledger Fabric version 2.0 beta has been released. While it has all fixes of 1.4.4 included and some interesting features like an improved chaincode upgrade process, it is not yet ready for production use. Additionally, the documentation and support of the beta are limited, increasing the difficulties during development. We, therefore, decided to use the stable release version 1.4.4.

2) Ordering Service: Hyperledger Fabric offers three different ordering services which are all plug-and-play: SOLO, Raft and, Kafka [28].

Solo only uses a single node that is in charge of the entire ordering service making it not crash fault-tolerant (CFT) [29].

This means that the crash of the single SOLO node would result in the crash of the entire network making it not suitable for production. Furthermore, this ordering service has been deprecated as of Hyperledger Fabric version 2.0 which is currently in beta.

Kafka is based on Apaches zookeeper [30], uses a "leader and follower" node configuration and is CFT. The network required by Kafka is more complex than Raft, mainly because it has more dependencies to deploy and maintain. Kafka has been deprecated as of HF version 2.0 as well.

Raft shares Kafka's advantages of being CFT without presenting its downsides, allowing for a more decentralized network with fewer dependencies. Furthermore, the official HLF documentation explicitly recommends using Raft which is additionally the only consensus algorithm implemented in version 2.0. For this reason, we chose Raft as our ordering service.

3) Chaincode programming language: Hyperledger Fabric does not require a domain-specific language for writing the smart contracts, we considered Java, Javascript and Go. While Java and Javascript SDKs are fully developed, Go is still under development.

We decided to use Javascript in combination with NodeJS due to its stable SDK, wide library support and wide adaptation.

4) Client application language: HLF's modularity is extended to the application language as well. We have the same choice of programming languages as in the previous section, the client application and the chaincode do not have to use the same programming language.

We decided to use Javascript for similar reasons as mentioned above, as a wider adaption in public projects and it's wide library support. Furthermore, it reduces the complexity of the project by not being forced to switch between languages while developing the application and the chaincode which are intertwined.

5) Number of organizations: We implemented two organizations. One representing the employees(TechCoEmployee) and one for the company(TechCoHR) which is represented by an employer of the human resources department. Making reference to the use case we wrote before, the only employee will be John and the only company manager will be Thomas.

6) Chaincode: The chaincode is used to implement the interactions between the client and the ledger, like adding of new content and updating or retrieving data, if the user is authorized to interact with said data.

The chaincode is distributed to every node and each node has to run the smart contract in order to update the world state, therefore we decided to make the chaincode as small and compact as possible, allowing for faster nodes, especially important for applications running on lower-end systems. This was achieved by moving most of the logic into the client applications. The implemented functions include:

- Issue: Creates a new asset, a contract, that will be added to the ledger when the employee accepts it. The contract contains the time of issue, the creator of the contract and information about the possible employee such as age or salary.
- AcceptContract: Once a contract for a possible employee exists, the employee can choose to accept it or deny it. If the contract is accepted, then the world state is updated, the contract owner changes to the new employee and the contract itself and the fact that the employee accepted it is added to the blockchain
- CheckContract: This function is used to pull the information of the contract from the ledger if the caller is authorized to view it. The function enters a comment stating why the information was requested. Both employer and employee are allowed to use this function, the employee however, is only allowed to look up his own contract.
- ModifyContract: This function is used to update information in the ledger, for example, an increased salary. Since this is just proof of concept, both parties can modify the data without the approval of the other. This should be addressed accordingly in a production environment.
- TerminateContract: If the employee decides to end his employment at the company, they invoke this function which returns all information on the blockchain about the current employee. The employee can then decide to request a deletion, anonymization of selected data. Due to legal reasons, some information has to be kept at the company, like the name of the departed employee. Note that in order to simplify this proof of concept, the employer is not allowed to terminate a contract.

7) Client application: We developed two different client applications since they require different specializations. The shared functionalities are:

- addToWallet: This allows the user to create an identity which allows them to interact with the network
- checkContract: An authorized user is given the ability to pull a contract from the ledger

The company, represented by the human resources department, has the following unique functionality:

- Issue: Creates the logic behind a contract, by using the chaincode function mentioned above.

The following functionality is dedicated to the employees:

- AcceptContract: It is an abstraction of the function named above, allowing an employee to accept an existing contract.
- TerminateContract: Only an employee with a valid contract is able to terminate their contract. It resets the ownership of said contract and allows the employee to decide how their data is handled after his departure from the company.

8) Number of channels: Our proof of concept uses one channel through which the employer and employee communicate the world state. It is possible to add more channels to allow for inter-company communication in order to create a fully documented employment history.

9) Number of peers and clients: We use two peers. One is in charge of the human resources department of the company client and the other one is in charge of the employee client application. Each peer runs an own client application to access the ledger.

10) Number of endorsers: Due to the small nature of our network, one endorser is sufficient to guarantee the liveliness of our system. This parameter should be revised and increased for a bigger network.

11) Privacy: Each access to the ledger is logged, alongside with the time and the user identification. Additionally, a message can be passed in which the user can justify the requested access. The result of every action, be it creating a new asset, or updating or reading an existing one, is logged to the blockchain making every interaction with the ledger traceable for both the employee and the employer.

VIII. Discussion

During and after the implementation of the proof of concept the following topics surfaces, which we think are important to mention:

A. Possible improvements

We identified the following improvements for our proof of concept that could be implemented in the future.

- Verify the scalability: Due to our limited resources we weren't able to verify that our solution is scalable enough for production use.
- Develop web apps: Currently, the interactions with the blockchain rely on the use of the command line, which is not intuitive for end-users. The system can be made more approachable by creating an easy to use web app. For programming languages, we would recommend using Vuejs if the client applications are written in Javascript,
- Including an existing Active directory: This is a milestone for the suitability of the system for production use because it would allow companies to avoid unwanted redundancy while making the integration into existing products easier.

B. Complex configuration of HLF

We encountered several problems while trying to use the basic Hyperledger Fabric network. The error logs were sadly not very clear and the help found online was very limited, and not as extensive as other blockchain frameworks. This is a serious drawback in comparison to other logging solution, with or without a blockchain component. The Hyperledger Greenhouse initially offered a simplified and easier way of deploying a HLF network using Hyperledger Composer. Unfortunately Composer has been deprecated as of August 2019 and does not support Hyperledger Fabric 1.4.4.

IX. Contribution

We achieved the following points:

- Analysis and evaluation of privacy requirements: We identified the private data that is generated and implemented our logging solution in a way that guarantees data protection from unauthorized access.
- Analysis of existing blockchain solutions: We compiled a list of the most promising blockchain solutions, using the best fitting consensus type for logging, Proof of Authority. Furthermore, we did an in-depth review of the three most promising blockchain solutions, and determined that Hyperledger Fabric is the best-equipped framework for us.
- Identification of a use case: We found multiple situations where a logging solution utilizing HLF could be adequately used. After careful consideration, we decided on the employer-employee situation because of the relationship between a company and its employees is not balanced. This means that the employee has been traditionally at a disadvantage when interacting with the company. For example, an employee could fear repercussions when asking the employer how their data is being handled. Our application bridges this gap.
- Implementation of the use case: After identifying the suitable use case mentioned above, we created a HLF network and implemented the chaincode to interact with the ledger and developed two applications for the different parties involved.

X. Related Work

Data logging is of such an importance that there is a large quantity of different logging solutions already available. Examples include Amazon CloudWatch [31] and Cloud Audit Logs [32]. However, not all of the existing solutions take full advantage of blockchain technology and some of them do not use it at all. The use of blockchain to log data is an ongoing topic of research in areas such as the improvement of logs for consulting health data in the European Union [33].

We found several proposals for using Hyperledger Fabric in a logging context. The two we found most interesting are 'Transparent Logging with Hyperledger Fabric' [34] and 'Logging mechanism for cross-organizational collaborations using Hyperledger Fabric' [35]:

The first paper [34] proposes to couple a private HLF blockchain to a public permissionless blockchain. The public blockchain serves as a trust anchor, while the private blockchain contains the sensitive logs, allowing the users to access their own data. Their use case involves many different parties that want to access the blockchain, therefore the authors focus on the performance of the network and test different configurations to find the one with the best throughput.

The second paper [35] aims to facilitate collaboration between different organizations by making the logging more transparent and secure. They use HLF to log the interactions of the different actors. The authors present a working prototype for their use case, utilizing Kafka as their consensus algorithm.

XI. Conclusion

Data collection and processing have become a lucrative business in the last years. After the Cambridge Analytica scandal revealed the power of data to the masses, data privacy has become a mainstream topic. Since then additional national and international laws enforce the data protection and privacy. It gives the individuals control over their own personal data. It states that "controllers and processors of personal data must put in place appropriate technical and organizational measures to implement data protection principles" [4].

Using our proof of concept, we offer a solution to address these issues in the employer-employee relationship by offering both parties the possibility to independently verify that the data usage is appropriate.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Cryptography Mailing list at https://metzdowd.com*, 03 2009.

[2] Largest number of banks to join live application of blockchain technology. [Online]. Available: https://www.jpmorgan.com/global/treasury-services/IIN

[3] What is dtube? [Online]. Available: https://about.d.tube/

[4] Gdpr. [Online]. Available: https://eur-lex.europa.eu/eli/reg/2016/679/oj

[5] Hyperledger fabric. [Online]. Available: https://www.hyperledger.org/projects/fabric

[6] Hyperledger documentation - ledger. https://hyperledger-fabric.readthedocs.io/en/release-1.4/ledger/ledger.html.

[7] S. D. Angelis, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, and V. Sassone, "Pbft vs proof-of-authority: Applying the cap theorem to permissioned blockchain," *Italian Conference on Cyber Security*, 2017.

[8] J. A. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," *Advances in Cryptology - EUROCRYPT*, 2015.

[9] S. D. Angelis, "Assessing security and performances of consensus algorithms for permissioned blockchains," *ArXiv*, vol. abs/1805.03490, 2018.

[10] S. T. Piergiovanni, "Invited paper: On the characterization of blockchain consensus under incentives." *SSS*, pp. 1–15, 2019. [Online]. Available: https://academic.microsoft.com/paper/2989688139

[11] Announcing the first SHA1 collision. Google. [Online]. Available: https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html

[12] J. M. Truby, "Decarbonizing bitcoin: Law and policy choices for reducing the energy consumption of blockchain technologies and digital currencies," *Energy Research & Social Science*, 2018.

[13] P. Gaži, A. Kiayias, and A. Russell, "Stake-bleeding attacks on proof-of-stake blockchains," in *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, June 2018, pp. 85–92.

[14] Parity technologies. [Online]. Available: https://www.parity.io

[15] Aura. [Online]. Available: https://github.com/paritytech/parity/wiki/Aura

[16] Parity github. [Online]. Available: https://github.com/paritytech/parity-ethereum

[17] Parity description. [Online]. Available: https://www.parity.io/ethereum/#intro

[18] Ethereum documentation of smart contract languages. [Online]. Available: https://ethereum.org/developers/#smart-contract-languages

[19] P. Ekparinya, V. Gramoli, and G. Jourjon, "The attack of the clones against proof-of-authority," *ArXiv*, vol. abs/1902.10244, 2019.

[20] Go ethereum. [Online]. Available: https://geth.ethereum.org

[21] Geth github. [Online]. Available: https://github.com/ethereum/go-ethereum

[22] The linux foundation. [Online]. Available: https://www.linuxfoundation.org/projects/

[23] J. Sousa, A. N. Bessani, and M. Vukolic, "A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform," *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 51–58, 2017.

[24] Case study: Circulor achieves first-ever mine-to-manufacturer traceability of a conflict mineral with hyperledger fabric. https://www.hyperledger.org/resources/publications/tantalum-case-study.

[25] Case study: How walmart brought unprecedented transparency to the food supply chain with hyperledger fabric. https://www.hyperledger.org/resources/publications/walmart-case-study.

[26] Case study: Sony global education chooses hyperledger fabric for a next-generation credentials platform. https://www.hyperledger.org/wp-content/uploads/2017/12/Hyperledger_CaseStudy_Sony.pdf.

[27] Smart contracts. [Online]. Available: https://www.investopedia.com/terms/s/smart-contracts.asp

[28] Hyperledger documentation - ordering services. https://hyperledger-fabric.readthedocs.io/en/latest/orderer/ordering_service.html.

[29] Hyperledger documentation - solo. https://hyperledger-fabric.readthedocs.io/en/latest/orderer/ordering_service.html#ordering-service-implementations.

[30] N. Garg, *Apache Kafka*. Packt Publishing, 2013.

[31] Amazon cloudwatch documentation. [Online]. Available: https://docs.aws.amazon.com/cloudwatch/index.html

[32] Cloud audit logs. [Online]. Available: https://cloud.google.com/logging/docs/audit/

[33] V. C. Luigi Castaldo, "Blockchain-based logging for the cross-border exchange of ehealth data in europe," *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2018.

[34] C. Schaefer and C. Edman, "Transparent logging with hyperledger fabric," *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pp. 65–69, 2019.

[35] L. Van Hoye, P.-J. Maenhaut, T. Wauters, B. Volckaert, and F. De Turck, "Logging mechanism for cross-organizational collaborations using hyperledger fabric," *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2019.