Technische Universität Berlin

Department of Telecommunication Systems
Chair for Open Distributed Systems

Master Thesis

Design and Application of a Node Onboarding Process for Blockchain-based Open Data Ecosystems

Anton Altenbernd
Matriculation Number: 351344
anton.altenbernd@campus.tu-berlin.de

Supervisor
Fabian Kirstein, Fraunhofer FOKUS

Examiners
Prof. Dr. Manfred Hauswirth, TU Berlin
Prof. Dr. Ingo Weber, TU Berlin

February 19, 2021

Statutory Declaration

I hereby declare that the thesis submitted is my own, unaided work, completed without any unpermitted external help. Only the sources and resources listed were used.

The independent and unaided completion of the thesis is affirmed by affidavit:

Berlin, February 19, 2021

_____

Anton Altenbernd

Abstract

Today Open Data has gained popularity and its impact and relevancy has been shown in several studies. Notable Open Data ecosystems, such as the European Data Portal (EDP), host more than a million datasets and involve a wide range of stakeholders. Above all, Open Data lives from its reusability which is endangered by reliability, data quality and interoperability issues of current Open Data ecosystems. To overcome these drawbacks, research proposes the application of blockchain. For this, blockchain may solve reliability issues with regard to shared common state and a uniform infrastructure may reduce interoperability issues and barriers between involved stakeholders. In addition, the application of blockchain may increase the data quality by introducing rules when data is published through the blockchain. Open Data is readable without permission, but the right to write is reserved only for known data publishers and data providers. Therefore, new ones have to go through an authorization process to take part in a public permissioned blockchain-based Open Data ecosystem. However, current research does not consider how data publishers and data providers can practically join such an ecosystem. Especially, simple mechanisms with as little human interaction as possible are indispensable. We propose an onboarding process for nodes of a blockchain-based Open Data ecosystem. This includes a dedicated Public Key Infrastructure for a Blockchain-based Open Data Ecosystem (BODE-PKI) which adopts the certification of new nodes and the distribution of certificates and their revocation status across all involved nodes. For this, the certification is highly inspired by the Automatic Certificate Management Environment (ACME) protocol which provides usability with little human interaction. The BODE-PKI is based on a authorization database that is stored distributed across all nodes and provides transparency and accountability with regard to Open Data principles. We evaluate the presented approach with Hyperledger Besu as the underlying blockchain system against derived requirements and demonstrate the desired applicability and performance. We show that the presented approach can cope with real-world environments but at the same time complies with fundamental requirements. Finally, the presented approach can serve as a basis for practical onboarding of nodes in a blockchain-based Open Data ecosystem.

Zusammenfassung

Gegenwärtig hat Open Data an Popularität gewonnen und seine Wirkung und Relevanz wurde in mehreren Studien nachgewiesen. Nennenswerte Open Data Ökosysteme wie das European Data Portal (EDP) beherbergen mehr als eine Million Datensätze und beziehen eine Vielzahl von Interessengruppen ein. Open Data lebt insbesondere von seiner Wiederverwendbarkeit, die durch Probleme mit der Zuverlässigkeit, Datenqualität und Interoperabilität der aktuellen Open Data Ökosysteme gefährdet ist. Um diese Nachteile zu überwinden, schlägt die Forschung den Einsatz der Blockchain Technologie vor. Dazu kann Blockchain die Zuverlässigkeitsprobleme durch einen gemeinsamen Zustand lösen und eine einheitliche Infrastruktur bieten, um Interoperabilitätsprobleme und Barrieren zwischen den beteiligten Akteuren zu reduzieren. Darüber hinaus kann die Anwendung von Blockchain die Datenqualität erhöhen, indem Regeln eingeführt werden, wenn Daten über die Blockchain veröffentlicht werden. Open Data ist ohne Erlaubnis lesbar, aber das Recht zum Schreiben ist nur bekannten Datenherausgeber:innen und Datenanbieter:innen vorbehalten. Daher müssen neue einen Autorisierungsprozess durchlaufen, um an einem Open Data Ökosystem teilzunehmen, das auf einer Public Permissioned Blockchain basiert. Die aktuelle Forschung berücksichtigt jedoch nicht, wie Datenverleger:innen und Datenanbieter:innen praktisch einem solchen Ökosystem beitreten können. Insbesondere sind einfache Mechanismen mit möglichst wenig menschlicher Interaktion unabdingbar. Wir schlagen einen Onboarding Prozess für Knoten eines solchen Open Data Ökosystems vor. Dieser beinhaltet eine dedizierte Public Key Infrastruktur für ein Blockchain-basiertes Open Data Ökosystem (BODE-PKI), die die Zertifizierung neuer Knoten und die Verteilung von Zertifikaten und deren Status bezüglich ihrer Widerrufung an alle beteiligten Knoten übernimmt. Dabei ist die Zertifizierung stark an das Automatic Certificate Management Environment (ACME) Protokoll angelehnt, das eine Benutzerfreundlichkeit mit reduzierter menschlicher Interaktion bietet. Die BODE-PKI basiert auf einer Autorisierungsdatenbank, die über alle Knoten verteilt gespeichert wird und Transparenz und Rechenschaftspflicht im Hinblick auf Open Data Prinzipien bietet. Wir evaluieren den vorgestellten Ansatz mit Hyperledger Besu als zugrundeliegendem Blockchain System gegen abgeleitete Anforderungen und demonstrieren die gewünschte Anwendbarkeit und Performance. Wir

zeigen, dass der vorgestellte Ansatz in realen Umgebungen zurechtkommt und gleichzeitig die grundlegenden Anforderungen erfüllt. Schließlich kann der vorgestellte Ansatz als Grundlage für das Onboarding von Knoten in einem Blockchain-basierten Open Data Ökosystems im praktischen Sinne dienen.

Contents

1 Introduction **1**

1.1 Context 1
1.2 Objective 2
1.3 Methodology 3

2 Background **4**

2.1 Open Data 4
2.1.1 Linked Open Data 5
2.1.2 Open Data Ecosystem 6
2.2 Blockchain-based Open Data Ecosystem 7
2.3 Cryptographic Fundamentals 8
2.3.1 Symmetric-key Cryptography 8
2.3.2 Asymmetric-key Cryptography 8
2.3.3 Hashing 9
2.3.4 Digital Signatures 10
2.3.5 Digital Certificates 11
2.4 CAP Theorem 11
2.5 Blockchain Fundamentals 12
2.5.1 Blockchain System 12
2.5.2 Blockchain 12
2.5.3 Transactions 13
2.5.4 Blockchain Types 14
2.5.5 Consensus Finding 15
2.6 Public Key Infrastructure 19
2.6.1 Public-Key Infrastructure using X.509 19
2.6.2 Automatic Certificate Management Environment 22
2.6.3 Web of Trust 23
2.6.4 Domain Name Service 24
2.7 Hyperledger Besu 27

3 Analysis **30**
3.1 Problem Statement 30
3.2 Assumptions 31
3.3 Requirements 31
3.4 Analysis of Related Work 33
3.4.1 Certification 33
3.4.2 Distribution of Issued and Revoked Certificates 35

4 Design of the BODE-PKI **38**
4.1 System Architecture 38
4.2 Authorization Database 40
4.3 Replication 41
4.4 Onboarding 42
4.5 Revocation 43

5 Implementation of the BODE-PKI **44**
5.1 Authorization Primary and Replica 44
5.2 Besu Configuration and Connector 49

6 Evaluation **53**
6.1 Predefined Requirements 53
6.2 Applicability and Performance 55
6.2.1 Basic Functionality 55
6.2.2 Synchronization 58
6.2.3 Access Control 60

7 Conclusion and Outlook **64**

A Figures **66**

B Prototype **68**

Bibliography **73**

List of Figures

4.1 Architecture of the BODE-PKI 39

5.1 Sequence Diagram of the Onboarding Process 48
5.2 Sequence Diagram of the Access Control Adjustment 52

6.1 Synchronization over Time during Onboarding with Crashes 59
6.2 Synchronization over Time during Revocation with Crashes 59
6.3 Access Control during Onboarding 61
6.4 Access Control during Revocation 61
6.5 Access Control during Reactivation 62
6.6 Access Control during Revocation and Reactivation 62

A.1 Individual Synchronization over Time during Onboarding with Crashes . 66
A.2 Individual Synchronization over Time during Revocation with Crashes . . 67

List of Tables

3.1 Comparison of the Certification 35
3.2 Comparison of the Distribution of Issued and Revoked Certificates 37

6.1 Results of the Basic Functionality Test 57
6.2 Results of the Access Control Test 63

List of Listings

2.1 Enode Url Format 28

5.1 Example Certificate in the Allow-Map 45
5.2 Example Record in the Authorization Database 46
5.3 Genesis File for the Besu Network Configuration 49
5.4 Configuration File for each Besu Node 50

List of Acronyms

ACME	Automatic Certificate Management Environment
API	Application Programming Interface
BODE-PKI	Public Key Infrastructure for a Blockchain-based Open Data Ecosystem
CA	Certificate Authority
CoT	Chain of Trust
CLI	Command Line Interface
CRL	Certificate Revocation List
CSR	Certificate Signing Request
CT	Certificate Transparency
DANE	DNS-based Authentication of Named Entities
DNS	Domain Name Service
DNSSEC	Domain Name System Security Extensions
DV	Domain Validated
ECDSA	Elliptic Curve Digital Signature Algorithm
EDP	European Data Portal
EV	Extended Validation
GHOST	Greedy Heaviest Observed Subtree
HTTP	Hypertext Transport Protocol
IBFT	Istanbul Byzantine Fault Tolerance
ICANN	Internet Corperation for Assigned Names and Numbers

IV	Individual Validated
JSON-RPC	JavaScript Object Notation Remote Procedure Call
KSK	Key-Signing Key
OCSP	Online Certificate Status Protocol
OV	Organizational Validated
PBFT	Practical Byzantine Fault Tolerance
PGP	Pretty Good Privacy
PITM	Person-In-The-Middle
PKI	Public-Key Infrastructure
PKIX	Public-Key Infrastructure using X.509
PoA	Proof of Authority
PoB	Proof of Burn
PoS	Proof of Stake
PoW	Proof of Work
RDF	Resource Description Framework
RR	Resource Record
SPARQL	SPARQL Protocol And RDF Query Language
TLS	Transport Layer Security
URI	Uniform Resource Identifier
UUID	Universal Unique Identifier
Web PKI	Web Public-Key Infrastructure
WoT	Web of Trust
ZSK	Zone-Signing Key

1 Introduction

1.1 Context

Open Data describes the process of making data readable and reusable by others. To some extent, it gives citizens insight into the activities of public authorities. In addition, it increases the amount of data available to citizens to perceive their own community and develop new innovative services. Open Data makes public authorities more accountable and transparent and thereby enhances trust[1]. Solar et. al.[2] argue that the key to adding value to Open Data is to encourage citizens to reuse it. However, the provision does often not meet non-functional requirements to support this desired reusability of Open Data[3].

A noteworthy example for the provision of Open Data is the European Data Portal (EDP)[1]. The EDP provides a platform where metadata is published about datasets that are provided by European governments and institutions. It constitutes a central access point for searching and retrieving Open Data from countries of the European Union. For this, metadata is harvested from country portals, transformed and enriched by additional information[4]. The EDP as an Open Data ecosystem organizes data publishers and data providers hierarchically. For instance, data is harvested from GovData[2], GovData fetches data from Open Data Berlin[3] and Open Data Berlin retrieves data from data providers like the State Office for Health and Social Affairs Berlin[4]. Moreover, the Publications Office of the European Union manages the EDP and decides which data portals are included.

In case of the EDP, publishers often only store metadata and datasets remain with the providers. Effectively, the EDP acts as single point of access that is under centralized control and is vulnerable to a single point of failure. In case of a failure, metadata may become unavailable and inaccessible and thereby unreliable. Nevertheless, reliability

[1]European Data Portal. (2021, January 21). https://www.europeandataportal.eu
[2]GovData. (2021, January 21). Datenportal für Deutschland. https://www.govdata.de
[3]Open Data Berlin. (2021, January 21). https://daten.berlin.de
[4]State Office for Health and Social Affairs Berlin. (2021, January 21). https://www.berlin.de/lageso/

issues may have a sustainable impact on the reuse of Open Data[5]. In addition, individual data providers and data publishers often use different publication technologies, which may lead to poor interoperability between participants and bad quality of metadata and the data itself[6].

A proposed solution for these drawbacks is the application of blockchain. As a distributed and decentralized service architecture, blockchain seems to have excellent properties for Open Data ecosystems[7]. In general, there may be no single point of failure in a blockchain-based Open Data ecosystem and a shared common state between involved participants could solve availability issues of metadata. Moreover, a central virtual access point could be provided that is reachable through multiple access points and is under decentralized control. Interoperability and data quality issues within the ecosystem may be eliminated as participants would use a uniform infrastructure[8, Chapter 4]. Other issues may be solved by the immutability and the auditing functionality of the underlying data storage, such as providing data integrity and trusted provenance information, managing persistent identifiers or tracking of third-party enrichments.

1.2 Objective

By now, there are several studies in the field of Open Data with regard to blockchain technology. Tran et al.[9] proposed a tool for metadata registry generation and deployment in blockchain systems and Garcia et al.[10] studied the deployment of metadata with blockchain. Moreover, English et al.[11], Third et al.[12], Sopek et al.[13] and Le-Tuan et al.[14] evaluated how blockchain can improve the Resource Description Framework (RDF) and Linked Data environments. Yet, current research in the field of Open Data and blockchain exclude aspects that are relevant in practice. In particular, it is left open how new data providers and data publishers can easily and practically join a blockchain-based Open Data ecosystem and how information about new members is distributed across the ecosystem.

Open Data is readable without permission, but the right to write is reserved only for known data providers and data publishers. Consequently, the underlying blockchain should be publicly readable without authentication and be writable by data providers and data publishers only with permission. For simplicity and without loss of generality, it is assumed that data providers and data publishers of the ecosystem respectively operate a single node that is associated with an account in the blockchain system. New data providers and data

publishers have to go through an authorization process to take part in a public permissioned blockchain-based Open Data ecosystem.

Since actors are well-defined, Open Data ecosystems serve as a good frame to elaborate node onboarding strategies for decentralized networks with certain permissions, especially in case of permissioned blockchain systems. The objective of this thesis is to answer the question of how a new node can join a blockchain-based Open Data ecosystem while taking the existing governance and authentication structures of Open Data into account. From here, we make the hypothesis that a dedicated Public Key Infrastructure for a Blockchain-based Open Data Ecosystem (BODE-PKI) can manage the creation, distribution, and revocation of certificates across all participating nodes.

1.3 Methodology

Beyond the context and the objective of this thesis, chapter 2 provides a literature review of the theoretical background. This review includes Open Data and gives an overview of existing research in the field of Open Data with regard to blockchain technology. Further, the review examines cryptographic fundamentals, briefly introduces the CAP theorem and covers blockchain fundamentals. The subsequent section examines well-known Public-Key Infrastructures (PKIs) and focuses in particular on the certification and distribution of certificates and their revocation status. Finally, the last section provides an introduction to Hyperledger Besu. In chapter 3 the problem statement and the theoretical background are used to derive assumptions and define requirements for a BODE-PKI. Elements of known PKIs are reviewed and assessed with respect to their ability to meet the requirements. From there, a conceptional design of a BODE-PKI is developed in chapter 4 by deriving design choices on the basis of the analysis from chapter 3 and the theoretical background from chapter 2. Chapter 5 presents the implementation details of the prototype that implements the proposed BODE-PKI. Further, chapter 6 evaluates the design against the predefined requirements of chapter 3 and the prototype is tested to show that the designed BODE-PKI can cope with a real-world scenario. For this, tests are defined by the requirements of a real-world Open Data ecosystem like the EDP and performed to show the applicability and performance of the presented approach. Thereby, the evaluation shows whether the presented approach meets the described objective. Finally, the last chapter summarizes the thesis and potential future work is presented.

2 Background

This chapter presents the theoretical background relevant to this thesis. Section 2.1 presents the Open Data movement where principles and components are discussed. The subsequent section 2.2 presents issues of traditional Open Data ecosystems and discusses how the blockchain technology can be supportive in this domain. Further background is reviewed in later sections where section 2.3 examines cryptographic fundamentals, section 2.4 reviews the CAP theorem and section 2.5 presents blockchain fundamentals. Moreover, section 2.6 provides an overview over known PKIs and their elements. Finally, section 2.7 introduces Hyperledger Besu.

2.1 Open Data

Data are raw symbols and form the foundation of the wisdom hierarchy[15]. Representations of the wisdom hierarchy typically define information in terms of data, knowledge in terms of information and wisdom in terms of knowledge[16]. For instance, relations in data are understood and interpreted as information, patterns in information are understood and collected as knowledge and principles in knowledge are understood to gain wisdom[17]. The Open Data movement tries to bring knowledge or even wisdom to the people or as stated by Emer Coleman, former Chair of the Open Data Governance Board Ireland: "Open Data is moving from the tyranny of the expert to the wisdom of the crowd".

According to the Open Knowledge Foundation, the key features of openness are availability and access, reuse and redistribution and universal participation. Open Data becomes open knowledge when it is useful, usable and used. Therefore, Open Data is free to use, reuse and redistribute without any legal, technological and social restriction[18]. Open Data is provided by governments, non-profit organizations, businesses and citizens. Coordination and organization of several entities form an Open Data ecosystem. Often the collection and publication of data is done by other organizations who provide certain platforms where consumers can read data and transformers can process data and republish it. Ideally, Open

Data supports the idea of open government where administration is made transparent and accountable to the population. Such data is usually referred as public sector information or open government data. Moreover, it encourages engagement and participation, e.g., providing feedback on government activities, and invites to give published data a value, e.g., reusing Open Data in new innovations. To name just a few, Open Data has several benefits in different categories: Enhancement of trust in government, creation of new social services and improvement of data quality[19].

By now, more than 2600 Open Data portals exist[20]. There are several innovations that unlock the value of Open Data[21]. In addition, Open Data competitions, like the EU Datathon, are advertised "to promote and improve the Open Data made available" and "to stimulate innovative reuse of that data"[22]. The European Commission analyzed the benefits of Open Data according to market size and value added to the gross domestic product, number of jobs created, cost savings for the public sector and efficiency or productivity gains. They conclude that "The potential of Open Data is tremendous" but for realizing the potential the number of Open Data initiatives has to increase. Finally, they recommend to improve portal usability, machine readable data, release of public data and more[21].

2.1.1 Linked Open Data

Open Data is closely related to linked data. The intersection of both is referred as linked Open Data. Technically, linked data is published data that is machine-readable, links to other datasets and can be reversely linked by other datasets[23]. Uniform Resource Identifiers (URIs) are used to name linked data, Hypertext Transport Protocol (HTTP) is used to make linked data accessible, standards like the RDF are used to provide useful information and links to other URIs are included to discover other data. Linked data can be queried semantically by the SPARQL Protocol And RDF Query Language (SPARQL) and is essential to realize a web of data, also referred as semantic web[24]. RDF is a structured machine-readable standard to provide metadata[25]. The latter can be defined as data that describes other data in order to help to understand or use it[26]. According to Tim Berners-Lee, the quality of linked Open Data can be classified in a five star scheme where stars are gained in incremental order by: making data available on the web under an open license, making data available as machine-readable structured data, making data available in a non-proprietary format, identifying data by using RDF and SPARQL and providing context by linking to other data[24].

2.1.2 Open Data Ecosystem

To understand how Open Data ecosystems are organized, the EDP[1] as a notable Open Data ecosystem serves as the basis for further analysis. The EDP provides a central access point for metadata of datasets that are provided by European governments and institutions. In numbers, the EDP manages metadata of more than one million datasets that are harvested from around 80 data portals in 36 countries. Data providers and data publishers in the ecosystem are organized hierarchically. The EDP gets data from country data portals and those get their data from regional data portals. Regional data portals usually get their data from local data portals and those publish data provided by local institutions. The EDP is managed by the Publications Office of the European Union. To get harvested by the EDP, a form[2] on their website must be filled out. This may result in communication via mail, telephone or even a face-to-face meeting to check whether the data portal meets the requirements of the EDP. Afterwards, the harvesting process may be modified to pull data from the newly added data portal. Lower down in the hierarchy, memberships of data portals are managed independently. On this occasion the EDP has no further influence but it is planned to "progressively harvest additional data collected by regional, local and domain specific portals"[3].

From the reviewed background about Open Data and the inspection of the EDP, following assumptions for Open Data ecosystems are made: An Open Data ecosystem consists of data providers, data publishers, data consumers and data transformers. In general, data providers and data publishers can be understood as participants feeding the ecosystem with data, and data consumers and data transformers can be understood as participants using that data. Whereby, some entity exists, which controls access in who is allowed to feed the ecosystem with data and enables that this data can be found by anyone. Moreover, data providers and data publishers who are part of the ecosystem are considered to be known. For this, the identity of a new data provider or data publisher is verified in some out-of-band agreement. If an Open Data ecosystem is ordered hierarchically, the administrative right to manage who can feed the ecosystem with data might be delegated to subordinated data publishers.

[1] European Data Portal. (2021, January 21). https://www.europeandataportal.eu.
[2] EDP: Contact Form. (2021, January 21). https://www.europeandataportal.eu/en/feedback/form?type=2
[3] EDP: Be harvested by us. (2021, January 21). https://www.europeandataportal.eu/en/about/be-harvested-us

2.2 Blockchain-based Open Data Ecosystem

In traditional Open Data ecosystems, data is distributed but not replicated. Therefore, no redundancy is provided and a single point of failure may compromises the accessibility and reliability of data[8, Chapter 4]. In addition, traditional Open Data ecosystems have interoperability issues across different sources and lack in quality of metadata and the data itself[27]. Open Data ecosystems and blockchain systems follow similar principles with regard to organizational aspects. This includes, the promotion of transparency and equality in information knowledge and access[7]. With a blockchain-based approach, data is replicated across nodes, and since the infrastructure is decentralized, no single point of failure may occur[9]. In a blockchain-based Open Data ecosystem each participant may run a node and shares its data through the shared blockchain. Each participant keeps authority over its data but shared common state between participants may increase reliability, accessibility and availability. All participants share responsibilities and no single entity must guarantee the security and maintenance of the system[7]. The research company Gartner[28] predicts that the use of smart contracts will increase the quality of shared data through the blockchain. Smart contracts provide the rules for sharing data, and nodes of the blockchain system verify that these rules are met. A uniform infrastructure is provided in which interoperability issues may be removed between participants of the ecosystem[9]. A blockchain-based approach may increase trust in data by providing an infrastructure that provides high security standards[7]. Data that is published or changes to it are packed in a transaction and signed by the private key of its originator. The immutability of the blockchain make these transactions irrevocable. Therefore, blockchain can ensure data integrity and decrease the amount of possible threats to it[29]. The immutability property and auditing functionality of the blockchain may bring other opportunities that are relevant to Open Data ecosystems, such as providing trusted provenance information[30] or tracking of third party enrichments[31]. By now, there are several studies in the field of Open Data with regard to blockchain technology. Tran et al.[9] proposed a tool for metadata registry generation and deployment in blockchain systems and Garcia et al.[10] studied the deployment of metadata with blockchain. Moreover, English et al.[11], Third et al.[12], Sopek et al.[13] and Le-Tuan et al.[14] evaluated how blockchain can improve the RDF and Linked Data environments.

2.3 Cryptographic Fundamentals

Cryptography studies the secure communication via public channels. This includes authentication of communication partners, protection against data alteration, proofing data integrity and providing data privacy[32, Chapter 13]. Cryptographic procedures are the basis of each blockchain system and Public Key Infrastructure (PKI). To better understand both, this section provides an introduction to the fundamentals of cryptography, which serve as a solution to the problems described when interacting over a public channel. For this, this section introduces symmetric-key and asymmetric-key cryptography, fundamentals of cryptographic hashing, digital signatures and certificates.

2.3.1 Symmetric-key Cryptography

When using symmetric-key cryptography, the same secret key is used for both encryption and decryption of data. Nowadays, symmetric-key algorithms are very fast and secure. Especially, when large amounts of data must be exchanged, symmetric-key cryptography is the choice. However, when using symmetric-key cryptography, communication partners must exchange the secret key in before to establish a secure communication channel. To set this up, symmetric-key cryptography is not able to do this on its own. Back in the days, a separate channel had to be used to exchange the key for the first time. For this purpose, it was only possible to meet in person to exchange the key, as long as there was not a trusted third party to take care of the delivery. Moreover, symmetric-key cryptography provides bad scalability as for each pair of communication partners a separate key is required. In a fully connected network, the overall number of keys increases quadratically with the number of users. Furthermore, symmetric-key cryptography provides no non-repudiation where a message can be clearly assigned to an author[33, Chapter 6].

2.3.2 Asymmetric-key Cryptography

With the use of asymmetric-key cryptography, a pair of keys is generated. The private key is kept as a secret and the public key is made available to others. For encryption and decryption both keys are necessary. When the private key is used for encrypting the data only the public key can be used for decrypting the data and vice versa. In comparison to symmetric-key cryptography, asymmetric-key cryptography provides good

scalability as each communication entity requires only one key pair. Moreover, asymmetric-key cryptography provides non-repudiation by using digital signatures and requires no additional channel to establish a secure communication channel between communication partners. However, asymmetric-key cryptography is less efficient in terms of encryption and decryption. The foundation of asymmetric-key cryptography are one-way functions. A one-way function can be easily computed but a solution for the inverse is computational hard to find. The same counts for generated key pairs. The public key can easily derived from its private key but not the other way around. The security of asymmetric-key cryptography is also based on the length of the keys. The longer the keys are, the more difficult it becomes to deduce the private key from its public key. Nowadays, the integer factorization problem and the discrete logarithm problem are popular one-way functions which are practically used. The integer factorization problem is the basis of RSA, which is an asymmetric-key cryptosystem that includes key generation, encryption and decryption and digital signatures. The discrete logarithm problem is the basis of the Diffie-Hellman key exchange. It is a public-key protocol where both communication partners derive a random secret key over a public channel. This key can then be used for symmetric-key algorithms. Moreover, Elliptic-curve cryptography transfers the problem of the discrete logarithm on the algebraic structure of elliptic curves. Elliptic-curve cryptography has smaller keys but guarantees the same level of security compared to RSA or other asymmetric-key protocols. Elliptic-curve cryptography was adapted in several asymmetric-key protocols which were originally based on the discrete logarithm problem[33, Chapter 6].

2.3.3 Hashing

Hashing provides a deterministic mapping between a string with arbitrary length and a fixed-length hash value. A cryptographic hash function must have few collisions and huge impact in the output when small changes in the input occur. For this, it must be hard to find two inputs which have colliding hash values. Further, hash functions are one-way functions. It is easy to verify the hash value when the input is given and computational hard to construct the input when the hash value is given. Especially, hashing can be used to protect the integrity of data. It can serve as a fingerprint for data which then can be used to check for alterations. For instance, hashing can be used to protect passwords by storing the hash value rather then the clear text of the password. When a user submits its password to the system, the clear text of the password is hashed and compared to the stored hash value. Then the system never knows the password but can verify whether the

user is authorized[34]. To check the integrity of several distinct data inputs that belong together, a merkle tree is an efficient solution. A merkle tree is a binary tree where each leave represents the hash value of an input. The parent node represents the hash value of the concatenated values of its child nodes. Therefore, the root value protects the integrity of all inputs and a change in any value in the tree also results in a noticeable change in the root value. Additionally, a merkle tree provides an efficient proof of existence. To proof that an input is part of the merkle tree, a path from the leave to the root is required. Then, a persuasive evidence can be constructed by providing the hash values of each neighbor along the path, the input itself and the root value[35, Chapter 9].

2.3.4 Digital Signatures

A digital signature mainly solves two problems which occur when interacting through public channels or retrieving data from storage. On the one hand, it serves as proof that data has not been altered by an adversary. On the other hand, it proves that the originator has control over the private key which belongs to the claimed public key. A suite of recommended digital signature algorithms are specified in the Digital Signature Standard by the National Institute of Standards and Technology. This includes the Digital Signature Algorithm, the RSA-Signature and the Elliptic Curve Digital Signature Algorithm (ECDSA). In general, a signature algorithm consists of three parts: the key pair generation process, a signature generation process and a signature verification process. A signatory is the owner of the key pair and is the only entity authorized to digitally sign with it. The private key must remain secret to prevent that a different entity could generate fraudulent signatures. To reduce the amount of data that has to be signed, a message digest is created prior to signature generation by running an approved cryptographic hash function on the given input. To generate a signature the message digest is encrypted with the private key. The generated signature, the message and the hash function can then be used by the reader to verify the signature. For this, the reader decrypts the signature with the public key of the sender and runs the hash function on the message. Afterwards, the decrypted version of the signature is compared to the output of the hash function. If both are equal the signature is considered to be valid[36].

2.3.5 Digital Certificates

Asymmetric-key cryptosystems by itself cannot prove that the claimed public key belongs to a certain entity. Without linking a public key to an ownership, the entity behind the public key could be anyone. Then, a Person-In-The-Middle (PITM) attack is easy viable. In a PITM attack an adversary claims to be the communication partner and, if successful, is trusted by the corresponding communication partner. In the worst case, the adversary is able to retrieve confidential information or even generate signatures on behalf of the believed entity. To remove this issue, digital certificates are used to link a public key to an ownership. Typically, this link is created by a trusted third party. For this, the trusted third party verifies the link and creates the digital certificate. Afterwards, the digital certificate is signed by the trusted third party to ensure integrity of the certificate[37, Chapter 3].

2.4 CAP Theorem

Consensus finding is an essential component of any blockchain system. Therefore, it is valuable to understand how consensus is fundamentally formed in distributed systems. The CAP theorem helps to classify consensus algorithms and to make a choice for a blockchain-based Open Data ecosystem in this regard.

In a distributed system the view of the state mainly depends on the read protocol. If the read protocol always provides a consistent view of the system state, strong consistency is provided. Any conflicting view of the state is internally resolved by the write protocol and never recognized by the read protocol. Moreover, the system can provide eventual consistency where a conflicting view of the state is readable and eventually resolved by the write protocol[38]. According to the CAP-Theorem, the design of a distributed system has some trade-off between consistency, availability and partition-tolerance. Using the internet and thereby operating on unreliable communication channels can lead to network partitioning. Therefore, mostly the trade-off between consistency and availability is considered. A distinction is therefore made between AP- and CP-systems. In an AP-system availability is preferred to consistency and often eventual consistency is provided. In a CP-system consistency is preferred to availability and strong consistency is provided[39].

2.5 Blockchain Fundamentals

As a central element of the problem statement, blockchain technology must be examined in order to understand the effects that can arise when building a blockchain-based Open Data ecosystem. This section will introduce fundamentals of the blockchain technology. For this, it covers the general concept of a blockchain system, the basic structure of a blockchain, the content and the lifecycle of a transaction, the types in which blockchain systems are classified, the consensus finding in a blockchain system and the immutability property of the blockchain.

2.5.1 Blockchain System

According to Xu et. al.[8, Chapter 1] a blockchain system consists of three parts: a network, a data structure and a protocol. The network consists of a set of machines, also called nodes. The data structure, also called blockchain, is replicated across all nodes[4] in the blockchain system. The protocol defines a set of rules under which the blockchain system operates. This set of rules includes several aspects, such as rights, responsibilities, means of communication, verification and validation of transactions, mechanisms for appending a new block, incentive mechanisms, ensuring authorization and authentication. Following Rauchs et. al.[40] each blockchain system lives in a wider stakeholder ecosystem. Involved entities with varying influences are unified in a process called social consensus. The social consensus sets the rules for the network protocol. Finally, the social consensus is even able to change certain rules that were previously set in the network protocol. In some cases a change of the network protocol can lead to a fork where some stakeholders agree and some do not. If these changes split the blockchain system into two separate independent systems with no compatibility between them, it is referred as a hard fork. Otherwise, if there is still a compatibility, it is referred as a soft fork[41].

2.5.2 Blockchain

A distributed ledger is an append-only register for transactions which is replicated across multiple nodes. The state of the ledger is represented by contained transactions and their

[4] Throughout this thesis, nodes always refer to full nodes. As mentioned in 1.2, data providers and data publishers are considered to respectively operate a node replicating the blockchain in a blockchain-based Open Data ecosystem. Therefore, light nodes are neglected and not further mentioned in this thesis.

sequence[8, Chapter 1]. A blockchain is a distributed ledger where the way of integrating transactions in the ledger is defined in a specialized way. Typically, the blockchain is an ordered list of blocks where each block contains a finite list of transactions. Besides transactions, each block contains a block header. This block header carries in addition to other values an immutable representation of the transaction list. Usually, a merkle tree is therefore constructed from the transaction list and then the merkle root is included in the block header. Additionally, to order the blocks a block header contains the representation of the previous block. The representation of a block is the hash value of its block header. In order to manipulate a transaction, a change of the merkle root, the block hash value and all subsequent blocks is necessary because else the chain of hashes would automatically be invalidated. Thus, normally a block is more secure when other blocks follow. An essential block is the genesis block. The genesis block is the first block in the blockchain. It contains the initial state of the ledger and has no link to a previous block[42].

2.5.3 Transactions

Transactions are the main ingredient of a blockchain. Each newly appended transaction updates the current state of a blockchain. In general, transactions can hold arbitrary data but the network protocol defines the structure of a valid transaction. The initiator is linked to the transaction to provide integrity and authorization. For this, the initiator is associated with a public key and the transaction is digitally signed with the corresponding private key. After creating and signing a transaction, the initiator propagates the transaction to the network. If the transaction is valid, the transaction gets further propagated by nodes that already received the transaction. During and after propagation, the transaction is considered as pending. The network protocol defines how a pending transaction must be verified. Finally, a verified transaction may be added and eventually confirmed to the blockchain[8, Chapter 1]. In early blockchain systems, such as Bitcoin[43], transactions are mainly used to exchange a predetermined digital asset between accounts. Those blockchain system are referred as cryptocurrency systems[44]. In later blockchain systems, such as Namecoin[5], digital assets exists in form of a key-value pair and can be created or updated through transactions. Next generation blockchain systems, such as Ethereum[45], are able to store executable program code via transactions in the blockchain. To execute code, transactions carry function calls and input parameters. Then, the result is computed and verified by the nodes and possibly stored in the blockchain.

[5]Namecoin. (2021, January 21). https://namecoin.org

2.5.4 Blockchain Types

Depending on the network protocol, the participants of a blockchain system are allowed to either read or write the blockchain or do both. Writers are allowed to issue a transaction to update the blockchain state and readers are only allowed to see the contents of the blockchain[46]. In the following, known types of blockchain systems are listed. For simplicity, a single group of readers and a single group of writers is considered. Towards Welzel et al.[46] and Wüst et. al.[47], the classification can be divided into four types depending on the permission about the reading and the writing:

- In a public permissionless blockchain system everyone is considered equal and no trusted third party manages any membership. Everyone is free to join and leave the network and is allowed to read and write the blockchain. A blockchain system is designed as public permissionless if the state of the blockchain needs to be public verifiability and participants are in general unknown.
- In a public permissioned blockchain system the membership of the writers is managed by a trusted third party or a collection of privileged parties. Further, everyone is allowed to read the blockchain. A blockchain system is designed as public permissioned if the state of the blockchain needs to be public verifiability and the group of readers is unknown while the group of writers is known.
- In a private permissioned blockchain system the membership of all participants is managed by a trusted third party or a collection of privileged parties. Here the group of writers is a subgroup of the group of readers. A blockchain system is designed as private permissioned if the state of the blockchain is confidential, no public verifiability is necessary and all participants are known.
- In a private permissionless blockchain system the same properties as in the private permissioned blockchain system are fulfilled except that there is no distinction between the group of readers and the group of writers.

Beyond this broad overview, a finer-grained distinction can be made, for instance, by allowing only certain nodes to generate blocks while others are only allowed to issue transactions.

2.5.5 Consensus Finding

In a blockchain system different views of the system state may compete in multiple conflicting branches in the blockchain. Those occur if two or more proposed blocks reference to an identical predecessor[48]. To resolve those conflicts and find common agreement a consensus protocol is used. In most blockchain systems, the decision making of those is either final or probabilistic[49]. In the following, consensus algorithms of blockchain systems are studied and also classified according to the CAP-Theorem.

Nakamoto Consensus

The Nakamoto Consensus or longest chain rule is defined as part of the read protocol in a blockchain system. It defines a block as part of the current system state if the block is part of the longest branch and is followed by a parameterized number of successor blocks[43][50]. A blockchain system that uses the Nakamoto Consensus requires nodes to "prove the possession or commitment of certain measurable resources" in order to append a new block to the blockchain[51]. If one controls the majority of such resources a malicious manipulation of the system state is possible, which is also known as the 51% attack[52]. However, the overall security and applicability of the Nakamoto Consensus mainly depends on the resource type and how the proof is used in combination. Therefore, the following will go more into detail about a few resource types and their proof.

Proof of Work

In Proof of Work (PoW) based blockchain systems a block creator is called a miner. For block creation a miner has to prove the use of computing power. For instance, Bitcoin requires miner to find a nonce as part of the block header, so that the hash value of the block is smaller than a specified target value. As the hash value is unpredictable, finding the right nonce is a randomized process where the target value describes the difficulty of finding this nonce[53]. The block generation rate depends on the number of miners that are actively involved and the difficulty of the proof[54]. For this, the difficulty is dynamically set so that in average a constant block generation rate is given[53]. When PoW is combined with the Nakamoto Consensus, the block generation rate significantly influences how probable conflicting branches occur[54]. However, miner will always follow the longest chain as the probability is higher that their block will be part of the system state. Consequently, one

main branch will evolve over time and branches earlier in the chain become improbable[50]. If a blockchain system is partitioned, partitions may evolve their own longest chain and deliver a conflicting view of the state. Thus, as an AP-system the Nakamoto Consensus with PoW prefers availability over consistency and provides eventual consistency[55]. Yet, in a public permissionless blockchain system, such as Bitcoin or Ethereum, this case is fairly unrealistic as there are enough miners who are well connected. In contrast, in a permissioned blockchain system the number of nodes is relatively small and a partitioning is more likely. Consequently, if a stronger promise of consistency is required, an alternative consensus protocol may be preferable.

Proof of Stake

In Proof of Stake (PoS) based blockchain systems a block creator proves possession of a cryptocurrency[48]. There are several hybrid variations which combine PoS with PoW, e.g., Casper[56] and SnowWhite[57]. The PoS based blockchain system PeerCoin[6] allows miner to consume coin age and thereby reduce the difficulty of finding the right nonce. In some PoS algorithms, like Ouroboros[58], a leader is picked not only depending on the amount of stake but also depending on some random factor. A particular form of PoS is the Delegated Proof of Stake[59] where stake is used to vote for block creators. Some PoS algorithms are combined with the Nakamoto Consensus where conflicts are resolved eventually by the longest chain rule[48]. Those blockchain systems can be viewed as AP-systems. In contrast, Tendermint[7] or Algorand[8] finalize blocks immediately by using a non-probabilistic voting algorithm where the voting power is based on the amount of stake. Those blockchain systems can be viewed as CP-systems. However, a PoS based blockchain system can be manipulated when an adversary controls more than one third or one half of the stake depending on the PoS algorithm[60]. In permissioned blockchain systems usually the amount of nodes is relatively small. If the stake is improperly distributed among nodes the system becomes vulnerable to attacks where only a small amount of nodes needs to be compromised. In contrast, the amount of nodes in a public permissionless blockchain system is relatively high. Therefore, a compromise of a large amount of stake seems to be difficult. Nevertheless, it is no secret that in the Bitcoin network less than one percent of

[6] Peercoin. (2021, January 21). https://www.peercoin.net
[7] Tendermint. (2021, January 21). https://tendermint.com
[8] Algorand. (2021, January 21). https://www.algorand.com

all addresses are in control of the majority of all Bitcoins[9].

Proof of Authority

In Proof of Authority (PoA) based blockchain systems only an authorized group of nodes is allowed to create and validate new blocks. Especially, this can be realized in permissioned blockchain systems where nodes are known[61]. In the following, the PoA based consensus algorithms Aura[10] and Clique[11] are examined. Aura utilizes the Nakamoto Consensus and Clique is based on the Greedy Heaviest Observed Subtree (GHOST) protocol. In contrast to the Nakamoto Consensus, GHOST allows to include blocks that are not part of the main chain in the view of the system state by reference those in the main chain[45]. Both Aura and Clique select a leader out of the authorized group in a round robin fashion. De Angelis et. al.[61] argue that Aura and Clique based blockchain systems are classified as AP-systems. They reason that Aura has no consistency guaranty at all and Clique guarantees eventual consistency. Moreover, Ekparinya et. al.[62] have shown that Aura and Clique are vulnerable to a so called cloning attack where a node when selected as the leader can add a block twice to the blockchain by sending it in a time-delayed manner. In addition, there are also PoA-based blockchain systems that can be classified as CP-systems. These use the Practical Byzantine Fault Tolerance (PBFT) consensus algorithm or a variation of it.

Practical Byzantine Fault Tolerance

PBFT is a voting-based consensus algorithm that was introduced by M. Castro and B. Liskov in 1999. It guarantees safety and liveness as long as the number of faulty nodes F is less than one third of the total number of nodes N, $F < \frac{N-1}{3}$. To reach agreement upon all nodes, it uses a three phase commit protocol. The three phases are called pre-prepare, prepare and commit. At first, the client sends a request to the primary node. The primary node validates the request and broadcasts the pre-prepare message. Then the backup nodes validate the request contained in the pre-prepare message. If the request is valid they broadcast the prepare message. The prepare message tells the other nodes that the request is valid. Then all nodes collect prepare messages. If they received $2F$ prepare messages

[9] Bitcoin Rich List. (2021, January 21). BitInfoCharts. https://bitinfocharts.com/top-100-richest-bitcoin-addresses.html
[10] Parity. (2021, January 21). https://www.parity.io
[11] Clique. (2021, January 21). Ethereum. https://github.com/ethereum/EIPs/issues/225

they broadcast the commit message. After receiving $2F + 1$ commit messages, the request is viewed as globally valid. Afterwards a response is sent to the client. The client waits for $F + 1$ responses to make sure that the request is validated by all the nodes. In case that a primary node is faulty a new one is selected by using the view change protocol, which has a similar sequence as the commit protocol[63]. PBFT guarantees strong consistency by finalizing decisions immediately[64]. Therefore, a PBFT based blockchain system favors consistency over availability and can be classified as a CP-system. Moreover, PBFT scales badly as the number of messages rises quadratically with the number of nodes[64]. A consensus algorithm that is inspired by PBFT and focuses on blockchain systems is provided by the Istanbul Byzantine Fault Tolerance (IBFT) 2.0. In IBFT 2.0 a block creator can be viewed as both the client and the primary node. Blocks are created in rounds and the view change protocol is used to select a new block creator in a round robin fashion[65].

Immutability

The state of a blockchain system is said to be immutable but there might be moments where deletion of content is desirable. In reality the immutability of a blockchain mainly depends on the consensus algorithm. As mentioned in above sections, the Nakamoto Consensus provides probabilistic immutability as the blockchain is rewritable if the majority of the resource is under control. Other consensus algorithms that are not based on the longest chain rule finalize a new block directly and therefore have a stronger immutability guarantee. However, a blockchain can always be rewritten by social consensus, as mentioned in section 2.5.1. For this, members of the blockchain system can collectively decide to reset the point of truth to a specific block in the blockchain. Another option is to logically delete content. Proof of Burn (PoB) is a mechanism to make an asset inaccessible in the blockchain system. For a PoB, the asset is transferred to an unused address and therefore loosing any affiliation[66]. Alternatively, an asset could be flagged as deleted by a dedicated transaction. Then, still an affiliation is given and in contrast to the PoB mechanism a reversal is possible. However, in both ways, the content is still available to readers and must be interpreted as nonexistent.

2.6 Public Key Infrastructure

To design a BODE-PKI, well-known PKIs are studied. Throughout this section, the focus is mainly on onboarding techniques and how certificates and information about their revocation status are distributed. At first, the commonly used model Public-Key Infrastructure using X.509 (PKIX) is examined. Then, the Automatic Certificate Management Environment (ACME) protocol as an onboarding technique and the Web of Trust (WoT) as an alternative model are studied. Finally, the capabilities of the Domain Name Service (DNS) to support a BODE-PKI are investigated.

2.6.1 Public-Key Infrastructure using X.509

In the PKIX model certificates are always issued and revoked by a authorized trusted third party, also referred as the Certificate Authority (CA). A certificate is signed by a CA and eventually revoked for different reasons through various mechanisms. The basic purpose of a certificate is to bind a subject to a certain public key with a limited valid lifetime[67]. In the following important concepts regarding the PKIX model are studied.

Certificate Format

The X.509 standard gives a format for describing certificates. The basic format includes following fields: *version*, *serial number*, *signature algorithm*, *signature value*, *validity*, *issuer*, *subject* and *subject public key info*. The *validity* field always describes a certain time interval for which a certificate is valid. The *issuer* and *subject* field includes information about both entities respectively. The *subject public key info* field identifies the public key and the algorithm in use. Optional fields are *unique identifiers*[12] and *extensions*[13]. The field *unique identifiers* can be used to define a distinguishable name for issuer and subject that may be used across multiple certificates. The *extensions* field is used to give additional information about the issuer and subject, managing relationships between CAs and define certain rules which are used for revocation[67].

[12] Only available in version 2 and 3
[13] Only available in version 3

Certificate Types

In general, certificates are distinguished in end entity certificates and CA certificates. In contrast to the subject of a CA certificate, the subject of an end entity certificate is not authorized to issue certificates. CA certificates are divided into three types: cross-certificates, self-issued certificates, and self-signed certificates. In the case of cross-certificates, the issuer and the subject belong to different entities, while in the case of self-issued certificates the issuer and the subject belong to the same entity. Cross-certificates are especially used to describe a trust-relation between two entities. Self-signed certificates are a subgroup of self-issued certificates where the public key included in the certificate is also used to sign the certificate[67].

Chain of Certificates

To obtain the validity of a certificate a user needs to have a assured copy of the public key that signed the certificate. If the public key is not known an additional certificate is needed to obtain the validity of that public key. Therefore, a chain of certificates is provided where the last certificate in the chain is signed by an assured public key. The latter is always classified as a self-signed certificate and also called the root certificate[67].

Revocation

Besides expiration of validity there exist several reasons for a certificate to become invalid. One reason is to renew information associated with the certificate. Therefore, a CA issues a new certificate and revokes the old one. Another reason is when malicious behavior is exposed, e.g., when the private key of the subject got compromised. The X.509 standard proposes one mechanism for revoking certificates. Therefore, each CA holds a Certificate Revocation List (CRL) in a public repository. This CRL is periodically updated and signed by the CA. To check if a certificate is valid and not revoked the client requests the CRL and checks if the *serial number* is contained. To tell the client where to find the CRL, an entry in the *extension* field is added[67]. However, in reality there are two major problems which occur when using a CRL. On the one hand, a CRL can get very large and it becomes inefficient to go through the entire list. On the other hand, CRLs are updated periodically every hour, day, week or even monthly. Therefore, the user possibly

will notice a revoked certificate too late[68]. Another solution is defined by the Online Certificate Status Protocol (OCSP). To check if a certificate is valid and not revoked, the client sends a OCSP request to the OCSP responder which is usually maintained by the CA. Depending on the state, it then sends an answer containing *good*, *revoked* or *unknown* and a validity interval[69]. Ideally the OCSP responder verifies the state in real-time by checking the database of the CA. In addition, the OCSP response is signed by the OCSP responder itself or the associated CA[68]. In order to reduce the communication effort, the OCSP stapling standard can be used. For this, the certificate status request extension in the Transport Layer Security (TLS) handshake enables the certificate holder to attach an OCSP response that is provided with a time stamp and signed by the corresponding CA[70].

Web Public-Key Infrastructure

In practice, the PKIX model is widely used in the Web Public-Key Infrastructure (Web PKI) that involves web browsers, web servers and CAs. When a browser wants to establish a secure connection to a server, a TLS handshake is normally performed. Therefore, the TLS handshake contains a certificate provided by the server and issued by a CA[71]. Usually, each browser holds a preinstalled list of root certificates which is maintained independently. Therefore, multiple root CAs provide the trust anchor. To protect itself, each root CA authorizes certification to multiple intermediate CAs. These intermediate CAs are authorized to issue certificates for any end entity. Together they form a Chain of Trust (CoT) where an end entity certificate can always be traced back to one root CA[72]. As stated by the Electronic Frontier Foundation[14] browsers trust a large number of CAs directly or indirectly. Together with CAs all big internet browser software vendors, like Google, Apple and Mozilla, form a consensus-driven organization called the CA/Browser forum[15]. Especially, in their baseline requirements they distinguish end entity certificates by Domain Validated (DV), Organizational Validated (OV), Individual Validated (IV) and Extended Validation (EV) and define how CAs should validate these certificates so that they can be classified as trustworthy by the browsers[73][74]. Delignat-Lavaud et al.[72] and Kumar et al.[75] have shown that the gap between guidelines and practice has become smaller over time but not totally vanished.

[14]Electronic Frontier Foundation SSL Observatory. (2021, January 21). Electronic Frontier Foundation. https://www.eff.org/observatory

[15]CA/Browser Forum. (2021, January 21). https://cabforum.org

Certificate Transparency

In 2011 DigiNotar, a former dutch CA, got compromised and multiple rogue certificates were issued[76]. Afterwards, DigiNotar was removed as a root CA in browsers. This and other breaches have been the motivation for Google's Certificate Transparency (CT). CT tries to make the certification process worldwide publicly available. In this regard, it tries to build upon the current infrastructure and augments the CoT. CT consists of three components of which several can exist: certificate logs, monitors and auditors. A certificate log is a network service that acts an append-only database for certificates where certificates are stored in a merkle tree. The good behavior of a certificate log can be verified by a log proof that is based on the structure of the underlying merkle tree. A monitor contacts a log server and watches for suspicious certificates. They report for mistakenly or maliciously issued certificates and verify that all added certificates are visible in the log. An auditor verifies that the log server is behaving properly. In particular, it can be checked whether a certificate has been added to the log. Basically, log servers and monitors can be maintained by anyone who is interested in improving CT. But auditors are usually implemented within a browser and therefore the browser decides which log servers are trusted. Chrome currently accepts 41 log servers which are maintained by Google and known CAs, like DigiCert[77]. To anticipate log compromising Chrome needs to verify that a certificate was added in at least two logs. In May 2020 one of DigiCert's log servers got compromised and afterwards placed in read-only mode[78].

2.6.2 Automatic Certificate Management Environment

The ACME protocol is used by the widespread non-profit CA Let's Encrypt[16]. In February 2020, they reported the billionth certificate that they issued using the ACME protocol. Especially, they make the ease of use responsible for the adoption and the success of the ACME protocol[79]. The ACME protocol describes a method to automatically prove control over a domain for the issuance of DV certificates in the Web PKI. Before the ACME protocol was introduced, methods were used which required user interaction and in many cases were difficult to use. In contrast, the request, verification and issuance of a certificate can be performed automatically without human intervention when using the ACME protocol. To do so, the client generates a new asymmetric key-pair and requests an account at the selected CA. For this, the client signs a nonce given by the server with its

[16]Let's Encrypt. (2021, January 21). https://letsencrypt.org

private key to prove control over its public key. After the registration is completed, the client can order authorization for a domain. Since there are many ways to prove control over a domain the server selects a set of challenges that can be met by the client. Then, the client responds with a set of responses that tell the server which challenges are satisfied. Afterwards, the server verifies that the challenges are indeed completed. In the HTTP challenge, a client must provide a server-specified string in a file under a predefined HTTP location under the domain, and in the DNS challenge, a client must store a server-specified string in the *TXT* record for the domain. After confirmation of success, the client is authorized to request a certificate for the domain. For this, the client makes a Certificate Signing Request (CSR) that contains the domain and a specified public key. The CSR is signed by the authorized private key and is sent to the server. Afterwards, the server issues the ordered certificate and sends it to the client[80].

2.6.3 Web of Trust

The idea of a WoT is closely related to OpenPGP. OpenPGP is a standard that uses asymmetric-key and symmetric-key cryptography to ensure confidentiality between communication partners. It also provides key management, authentication and digital signatures[81]. Well-known implementations include Phil Zimmermann's Pretty Good Privacy (PGP) and its open-source counterpart GnuPG/GPG[17]. In contrast to a CoT, users decide who is trusted and who is not. For this purpose, users can decide if another person's key is valid by signing it and adding it to their key-ring. In addition, users can specify the level of trust they place in other users when validating keys. For this, users can define unknown, none, marginal and full trust. From the user's point of view, any key that is validated by a fully trusted user, which is also referred as an introducer, or by a number of marginally trusted users is automatically seen as valid and set to unknown trust[82]. In GnuPG/GPG the default number is three[18]. In addition, users can also authorize others to act as a meta-introducer. For this, a user has to trust-sign the key of another user with a specified level n. Then, any other user that is $n-1$ hops away and trusted by the meta-introducer is automatically trusted by the user[81]. In this way, OpenPGP gives a user a great deal of flexibility and control in defining their own WoT. To distribute certificates and their revocation status, users of OpenPGP usually use key servers to upload certificates. These can be own certificates, signed certificates or revocation certificates. Certificates can

[17] GnuPG. (2021, January 21). https://www.gnupg.org

[18] Validating other keys on your public keyring. (2021, January 21). GnuPG: Manual. https://www.gnupg.org/gph/en/manual/x334.html

be requested on key servers by specifying the name, e-mail or fingerprint of the user being searched, where the fingerprint is a short form of the actual public key. Finally, content is only provided if a known identifier is requested[82].

2.6.4 Domain Name Service

The DNS is a distributed database for indexing domain names mapped to host information. It is build as a single-rooted tree where each node stands for a label. The root is labeled as a dot and to provide uniqueness no two sibling nodes are allowed to have the same label. A domain name is defined as an inverted path in the tree. Therefore, the domain namespace contains all inverted domain paths in the full tree. A domain is defined as a subtree of the full tree and each domain in the subtree is considered as part of that domain. These domains are referred as subdomains. Children of the root are called top level or first level domains. Children of a top level domain are called second level domains and so on. Information about the domain namespace are stored by name servers. Each name server has complete information about a part of the domain namespace, also known as the zone. Each name server is the authority for a zone or even multiple zones. A zone contains all the domain names and associated information that are included in the domain except for delegated subdomains. Therefore, the name server has information about which name server is authoritative for which delegated zone. The name server responsible for a delegated zone may be run by another organization. The client of a DNS is called a resolver and each name server can also act as a resolver for clients. Resolution starts at a name server for the root name space, also referred as root server. Each root server knows were the authoritative name servers for the top level domains are. Then a name server of the top level domain knows where the authoritative name servers for the second level domains are and so on. Resolution can be either proceeded iteratively or recursively. To reduce load on the root servers resolutions are cached. Each resolution has a time to live interval and can not be cached forever. Associated data is stored in the Resource Records (RRs). In the internet, DNS is mainly used for mapping domain names to Internet Protocol addresses. Currently, the internet DNS consists of 13 root servers[19]. In addition, each name server can be split into several physical name servers, which are then load balanced. The root servers are managed by the international non-profit multi-stakeholder community Internet Corperation for Assigned Names and Numbers (ICANN)[20] and top

[19]List of Root Servers. (2021, January 21). IANA. https://www.iana.org/domains/root/servers
[20]ICANN. (2021, January 21). https://www.icann.org/

level domains are delegated to other organizations. Therefore, the administration of the internet DNS is decentralized. However, each domain authority has the power to decide to whom a zone is delegated[83, Chapter 1 and 2]. There exist several RR types and in the following a few are mentioned: *A* is used for IPv4 addresses[84], *AAAA* is used for IPv6 addresses[85], *CAA* can be used to name acceptable CAs[86] and *CNAME* can provide an alias domain name[84]. *DNSKEY* is the key record used in the Domain Name System Security Extensions (DNSSEC), *DS* is the signing key and *RRSIG* is the signature used in the DNSSEC[87]. Moreover, the *TLSA* record is used to provide a certificate or public key associated with the corresponding domain name in the DNS-based Authentication of Named Entities (DANE)[88].

Domain Name System Security Extensions

The use of DNSSEC is intended to address security vulnerabilities in the DNS. In particular, the DNS is vulnerable to PITM attacks since DNS packages are originally not protected from third party modification. For this, DNSSEC provides data integrity and origin authentication[89]. In DNSSEC two key pairs are used: the Zone-Signing Key (ZSK) and the Key-Signing Key (KSK). The ZSK is used to sign RRs that are part of the corresponding zone. The public key of the ZSK and the public key of KSK are stored in a *DNSKEY* record that is afterwards signed by the private key of the KSK. A flag in the *DNSKEY* record helps to determine which *DNSKEY* record corresponds to the ZSK and which to the KSK. To provide a CoT the KSK is authorized by the parent zone. Therefore, the public key of the KSK is signed by the private key of the parent ZSK and stored in the parent *DS* record. Ideally, the identity of a delegated signer is proven in some out-of-band agreement. Then, the authenticity of the KSK can be traced back to the root servers over the *DNSKEY* and *DS* records of the parent zones. The public key of the root KSK is known and preconfigured in name servers which implement DNSSEC. The ZSK is often used and needs to be changed frequently. Therefore, a new ZSK is generated, the public key of the ZSK is stored in a *DNSKEY* record and signed the by the private key of the KSK. Afterwards, RRs can be resigned with the new ZSK. Thus, having two key pairs has the advantage that the parent zone does not have to be informed for the rollover of the ZSK. A rollover of the KSK is less common, but this requires the parent zone to change the *DS* record[83]. The ICANN ensures that no single entity has unilateral jurisdiction over the root KSK. The root ZSK rollover is performed in a publicly audited Root KSK Ceremony[21]

[21] DNSSEC. (2021, January 21). IANA. https://www.iana.org/dnssec

that takes place every four months. These ceremonies can also involve generating and replacing of the root KSK. Any Root KSK Ceremony always requires a quorum of trusted community representatives physically present[90]. However, the legitimacy of ICANN and the role of certain entities involved seems to be a science in itself[91][92]. Currently, DNSSEC is not widely adopted. For example, the ".com" top level domain has about 1.5% signed domains[93] and also the DENIC[22] states in their monthly report that about 1.5% domains provide DNSSEC in the ".de" top level domain[94]. However, the support of DNSSEC mainly depends on the domain name registrars[95]. Roth et al.[96] argue that the support of registrars improved but domain owners often fail to upload their *DS* record correctly.

DNS-based Authentication of Named Entities

The use of DANE makes it possible to store a certificate or the associated public key in the DNS. To develop the full potential of DANE it should be combined with DNSSEC. Therefore, the provided *TLSA* record should be signed and verifiable through the CoT provided by DNSSEC. The *TLSA* record is divided into four parts: the certificate usage field, the selector field, the matching type field and the certificate association data. The usage field currently distinguishes between four cases. When the usage field is declared as PKIX-TA or PKIX-EE a certificate is provided that needs to be validated in the certification path of the Web PKI. The PKIX-TA tells the client that the provided certificate is a CA certificate or a public key associated with a CA. Therefore, it constraints which CA in Web PKI is allowed to issue a certificate for that domain. Thus, using DANE in the Web PKI would prevent any CA from issuing certificates for any domain. The PKIX-EE tells the client that the provided certificate is an end entity certificate or a public key associated with that domain. When the usage field is declared as DANE-TA a trust anchor can be asserted. Therefore, the certificate or the associated public key of that trust anchor is stored. When the usage field is declared as DANE-EE the domain owner usually provides a self-signed end entity certificate for that domain. The selector field defines if the presented data is either the full certificate or just the *subject public key info*. The matching type field tells if the data is presented raw, hashed by SHA-256 or SHA-512. Originally, DANE is defined for TLS communication and therefore provides X.509 certificates[88]. Analogously, a RR was proposed for using DANE for OpenPGP[97]. DANE has the advantage that the revocation of a certificate is implicit when the *TLSA* record is removed

[22]DENIC. (2021, January 21). https://www.denic.de

or overwritten[98]. However, worse than DNSSEC, DANE is only sparse supported, e.g., by domain registrars in the ".de" domain[23].

2.7 Hyperledger Besu

This section introduces Hyperledger Besu, which is used as the underlying blockchain system for the prototype of this thesis. All information for this section, unless otherwise noted, is taken from the Hyperledger Besu documentation[24]. In particular, this section provides an overview and explores how Besu must be configured to meet the derived requirements of chapter 3. In addition, it investigates what elements are important to configure the access control of the underlying blockchain system.

Hyperledger Besu is a Java-based Ethereum client under the umbrella project Hyperledger[25], which is hosted by the Linux Foundation[26]. The implementation of Besu is open source and developed under the Apache 2.0 license. In addition, Besu provides flexibility in configuration domains, such as the consensus mechanism and how access is controlled. Besu implements Ethash, Clique and IBFT 2.0. The first is a Proof of Work and the latter are Proof of Authority based consensus algorithms. Moreover, a Besu network can be configured as public permissionless, public permissioned, private permissionless or private permissioned. For signatures, Besu uses the ECDSA with *secp256k1* as the elliptic curve parameters.

To configure Besu as a custom network, a genesis file must be created and located at each node. The genesis file contains network configuration items and genesis block parameters. In particular, it determines which consensus algorithm is used. Moreover, it defines the incentive for new blocks through the *blockreward* parameter and the *gasLimit* parameter defines the maximum gas for all transactions in a block. Gas can also be removed from consideration by setting the *gasLimit* parameter to the highest possible value.

Further configurations can be made in the configuration file of each Besu node. For the access control configuration, Besu distinguishes between the *nodes-allowlist* and

[23] Hoster und Registrare mit DNSSEC-Diensten. (2021, January 21). Heise Online. https://www.heise.de/ct/artikel/Hoster-und-Registrare-mit-DNSSEC-Diensten-2643530.html
[24] Hyperledger Besu Documentation. (2021, January 21). https://besu.hyperledger.org
[25] Hyperledger. (2021, January 21). https://www.hyperledger.org
[26] The Linux Foundation. (2021, January 21). https://www.linuxfoundation.org

accounts-allowlist. The first contains all nodes that are authorized to propose new blocks and participate in the consensus mechanism, and the latter contains all accounts that are authorized to issue transactions. For the *nodes-allowlist*, nodes are listed in the enode url format and for the *accounts-allowlist*, accounts are listed by the account address. The enode url format is depicted in listing 2.1 and is used by Besu to identify nodes. For the construction of the account address, the last 20 bytes of the corresponding public key are hashed with the Keccak-256 algorithm.

```
enode://{public key, excluding the leading 0x}:{ip}:{port}
```

Listing 2.1: Enode Url Format

Additionally, a Besu node must have a key pair to operate in the blockchain system. If no key pair exists, the key pair is initially generated at startup. However, the key pair can also be generated using the Besu command line tools and must then be stored in the location specified in the configuration file.

For running, maintaining, debugging and monitoring, Besu provides an Application Programming Interface (API) based on JavaScript Object Notation Remote Procedure Calls (JSON-RPCs). This API is accessible via HTTP and WebSocket. Moreover, the API is divided into multiple areas that can be activated per communication method. For this thesis, the APIs ADMIN, PERM, ETH, NET, IBFT and TXPOOL are relevant. The general purpose of the ADMIN API is the maintenance of node connections. The ADMIN API can be used to set up and take down connections to other nodes. The general purpose of the PERM API is the maintenance of account and node permissions. The PERM API can be used to add and remove nodes in the *nodes-allowlist* and accounts in the *accounts-allowlist*. The ETH API provides methods to read the blockchain and to issue transactions. The NET API can be used to check with how many nodes a node is connected. Moreover, the IBFT API can be used to configure which nodes act as validators when Besu is configured to use IBFT 2.0. Finally, the TXPOOL API can be used to observe the transactions pool and check whether certain transactions are included or not.

The APIs ETH, NET and TXPOOL must be publicly available for monitoring and to make the blockchain publicly readable. The ADMIN, PERM and IBFT API must be protected to prevent configuration from outside. Unfortunately, Besu does not provide a way to

selectively protect different APIs independently per communication method.[27] On the one hand, user permissions can be defined for different APIs. Then, user credentials are required for the publicly available APIs and others are required for the protected APIs. Alternatively, the publicly available APIs can be activated via HTTP and the others via WebSocket. Consequently, communication must then be protected via WebSocket and open via HTTP.

[27] Hyperledger Besu Chat. (2021, January 21). https://chat.hyperledger.org/channel/besu?msg=hwMCkpw4xZySHs2zQ

3 Analysis

In this chapter, based on the presented background, the actual problem statement is derived. From here, assumptions are made, and both the problem statement and the assumptions are used to define requirements. In addition, the assumptions section defines roles, which are also referred in subsequent chapters. Finally, elements of known PKIs and their ability to meet these requirements are analyzed.

3.1 Problem Statement

Derived from chapter 2.1, an Open Data ecosystem is only as much worth as its participants in respect to quality and quantity. To grow the ecosystem, new participants should have a low barrier in both technical and cost-effective manner of joining. For this, it should be easy to maintain memberships of the ecosystem. Moreover, administration of an Open Data ecosystem should be accountable and transparent to the public. Otherwise, it would contradict with the original idea of Open Data. Chapter 2.2 has embraced the benefits of a blockchain-based Open Data ecosystem. In this a key pair and an endpoint is necessary to operate a node. The key pair is used to authenticate action and the endpoint is used to authenticate location of a node. As only known data providers and data publishers are authorized, a link to the nodes identity is required. As mentioned in chapter 2.3.5 this link is referred as a digital certificate. To receive a certificate, the identity of the node must be verified, and the public key and endpoint of the node requires a proof of control. In addition to certification, certificates and their revocation status must be distributed across all participating nodes. Therefore, a Public Key Infrastructure for a Blockchain-based Open Data Ecosystem (BODE-PKI) is required that manages certification and distribution of issued and revoked certificates. Moreover, this BODE-PKI has to provide a dedicated onboarding process for new nodes, which is easy to use, has no costs and does not generate any technical boundaries. Finally, the onboarding process should require as little human interaction as possible.

3.2 Assumptions

This section lists the assumptions made during the analysis. They are especially derived from the chapters 2.1, 2.2 and 2.5.

- Open Data must be publicly available to everyone, but is only provided by known entities. Therefore, a public permissioned blockchain system is assumed, where the nodes are most likely operated by data providers and data publishers. For simplicity and without loss of generality, it is assumed that data providers and data publishers respectively operate a single node and have one account in the blockchain system. Therefore, these data providers and data publishers are referred to as Node Operators.
- A blockchain-based Open Data ecosystem is supervised by a Network Authority. The Network Authority is a centralized body that decides who participates in the ecosystem. The Network Authority is the only entity allowed to add new nodes by signing them a certificate or remove nodes by revoking their certificate. It should be noted that the Network Authority has no influence or control over data provided by others and stored in the blockchain. The Network Authority is in charge to provide access but the responsibility of maintaining the underlying blockchain system is given to the Node Operators.
- In blockchain-based Open Data ecosystems a low write latency is not a concern. A consensus algorithm is assumed that guarantees strong consistency where the finality of written data reinforces trust in Open Data. Therefore, according to the state of the art, it is assumed that the blockchain system can handle at most $F = \frac{N-1}{3}$ nodes that are subjects to a byzantine fault.

3.3 Requirements

This section lists the requirements for a BODE-PKI, based on the problem statement, assumptions, and prepared background.

Identity Verification: As in traditional Open Data ecosystems, there must exist some out-of-band communication to verify the identity of a Node Operator. Since the identity must be linked to the endpoint of the node, the endpoint of the node itself cannot be used to verify the identity. Therefore, out-of-band communication refers to any communication

where the endpoint of the node is not involved. Ideally, identity verification is done in a face-to-face meeting. In some cases verification of identity might have lower requirements and communication over telephone or e-mail might be sufficient. However, the type of out-of-band communication mainly depends on the policy of the Network Authority regarding their security requirements.

Proof of Control: During verification of the identity, the public key and endpoint must be exchanged in order to link them to the identity. Afterwards, a proof of control must be performed to ensure that the public key and endpoint are technically usable and correct. As seen in chapter 2.6.2 this process can be handled automatically.

Barrier to Join: As in traditional Open Data ecosystems, there must be no limitation to growth. Therefore, any barrier to join must be low in both technical and cost-effective manner.

Uniform Cryptosystem: Blockchain systems usually presume a single cryptosystem. It must be ensured that all authorized nodes use key pairs from one uniform cryptosystem.

Independence: On the one hand, a BODE-PKI or other services must not influence or endanger the ability of a blockchain node to operate on its own in any environment. On the other hand, the state of a certificate, regarding the revocation status, must be verifiable without requesting a trusted third party. Otherwise, the verification of transactions may be blocked if the trusted third party is not reachable.

Transparency and Accountability: In a blockchain-based Open Data ecosystem, the set of authorized nodes may change over time. In line with Open Data principles, changes to this set should be handled in a transparent and accountable manner. Thereby, nodes can be convinced that they see the same as others and unintentional changes due to misbehavior are detectable.

Modularity: Blockchain systems differ in many aspects and there is no single best solution for all domains. Therefore, a BODE-PKI must provide modularity in terms of the underlying blockchain system.

3.4 Analysis of Related Work

According to U. Maurer[99], a PKI consists of several components which are involved in issuing, revoking, storing and distributing of certificates. A PKI is a distributed database that stores certificates and additional information such as the revocation status. Moreover, it provides a mechanism to retrieve a certificate and additional information. In the following, we revisit the PKIs presented in chapter 2.6 in terms of the requirements that a BODE-PKI should meet. For this, the certification and the distribution of certificates are reviewed separately.

3.4.1 Certification

To develop a certification strategy, this section discusses certification in the aforementioned PKIs. This includes a review of the WoT, CoT, Web PKI and DNS/DNSSEC.

Web of Trust: PGP's WoT provides a PKI where each participant decides who is trusted and who is not. For this, the WoT provides no identity verification and no proof of control mechanism. As mentioned in chapter 2.6.3, to classify other participants a distinction is made between *validity* and *trust*. Former means that the key of the participant is validated and latter means that the participant is trusted to validate the identities of others. A key is valid when either personally signed, signed by a fully trusted key or signed by three marginally trusted keys. From the point of view of the Network Authority all nodes that are fully validated and at least set to unknown trust are authorized to feed the ecosystem with data. In practice, marginally trust should be ignored as relying on the verification by three marginally trusted keys would have a high barrier to join for new nodes. Finally, the Network Authority has full control on which keys are accepted to guarantee a uniform cryptosystem.

Chain of Trust: The Network Authority could act as a single rooted CA signing certificates for new nodes. Then, every node with a signed certificate is allowed to feed the ecosystem with data. Equally to the WoT, no identity verification and no proof of control mechanism is provided. Finally, the Network Authority has full control to guarantee a uniform cryptosystem.

Web PKI: Data providers and data publishers may have certificates that are valid in the Web PKI. Therefore, it might be reasonable to use these certificates. As mentioned in chapter 2.6.1 certificates are distinguished into DV, OV, IV and EV. IV certificates are

neglected here because data providers and data publishers are often organizations, not individuals. OV and EV certificates require verification of the real-world identity and DV certificates require just a proof of domain control. OV and EV certificates could be used to delegate the effort for identity verification to CAs. Some data providers and data publishers might already have an OV or EV certificate and some might have a DV certificate. However, OV and EV certificates cost money and thereby increase the barrier to join. DV certificates provide a link between the public key and the domain. A domain can be representative for an endpoint. Therefore, DV certificates could be used to delegate the proof of control mechanism to CAs. In addition, when using Web PKI certificates, information about the revocation status from the CAs point of view must also be taken into account. Moreover, Web PKI certificates allow the usage of a variety of cryptosystems and a uniform cryptosystem is hardly enforceable. In the end, Web PKI certificates are located at web servers and are intended for that usage. Therefore, either the node would require to be equipped with the private key of the certificate or operate under the same physical location where both would contradict with the requirement of independence.

DNS/DNSSEC: Data providers and data publishers could use the DNS to provide a public key in the *DNSKEY* record or a certificate in the *TLSA* record. Both are then directly connected to their domain. Using the DNS would not solve identity verification, but a proof of control mechanism for the domain would be implicit. However, those records are not secured without using the DNSSEC. As mentioned in chapter 2.6.4, the DNSSEC is widely unsupported. Therefore, requiring a domain with DNSSEC support would increase the barrier to join drastically.

The comparison of certification is summarized in table 3.1 where requirements are ordered by occurrence and not by importance. The use of DNS with DNSSEC or OV/EV certificates can be ruled out as the barrier to join is too high. When it comes to DV certificates, the dependency and not providing a uniform cryptosystem outweighs the advantages. Both the WoT and CoT can fulfill important requirements, but in both cases an identity verification and a proof of control mechanism must be implemented. However, the WoT model provides more flexibility and should therefore be preferred to provide a shape-able PKI. In conclusion, a PKI that is inspired by the WoT brings the desired freedom to meet the requirements. A proof of control mechanism that is based on the ACME protocol will be elaborated and the identity verification will be handled out-of-band between the respective Node Operator and the Network Authority.

Requirement	WoT	CoT	OV/EV	DV	DNS/DNSSEC
Identity Verification	no	no	yes	no	no
Proof of Control	no	no	yes	yes	yes
Barrier to Join	low	low	high	low	high
Uniform Cryptosystem	yes	yes	no	no	no
Independence	yes	yes	no	no	yes

Table 3.1: Comparison of the Certification

3.4.2 Distribution of Issued and Revoked Certificates

In addition to certification, a consistent view of issued and revoked certificates is required. To develop a distribution strategy, this section discusses the distribution of information in mentioned PKIs. This includes a review of the OCSP, CT, DNS, and key servers.

OCSP: When using OCSP to request the status of a certificate, it can be requested in different ways. On the one hand, it can be pulled directly before write operations are verified. Consequently, the verification may be delayed if the OCSP server is not reachable. Intentionally ignoring the fact that there might be an update can be an unsatisfactory solution. Thereby, this solution provides no independence. On the other hand, nodes can request the OCSP server on a regular basis. This is done either by requesting the revocation status of all other nodes that are believed to be authorized or by requesting the own revocation status. The latter would require nodes to provide their time stamped revocation status via OCSP stapling while communicating. However, depending on the period this would generate additional network traffic and a consistent view may propagate slowly. OCSP servers are not transparent because they lack a global view. In addition, they are not accountable, as compromised information can be added or deleted without logging these actions. Moreover, a compromised OCSP server could respond with equivocate answers to different nodes and supply an inconsistent view about authorized nodes in the ecosystem.

Certificate Transparency: Requiring certificates to be existent in a publicly available, verifiable and append-only database provides a better security. Here, malicious behavior can be detected by domain owners in the long run. CT provides a way to distribute certificates for domains in a transparent and accountable manner. Certificates are not remove-able and a

client can prove that its view is identical to the view of other clients. However, information needs to be pulled and as in OCSP a trade-off between propagation and dependency exists. Moreover, CT provides no information about revoked certificates.

DNS: The Network Authority could use the *TXT* record to provide issued and revoked certificates. This *TXT* record would then be pulled by the nodes. If the information exceeds the size of the *TXT* record, the *TXT* record may contain a hash of the information and the information itself may be provided under the domain of the Network Authority via HTTP. Ideally, the Network Authority would have a ZSK that is used to sign the *TXT* record and a KSK that is used to sign the ZSK. The KSK itself is signed by the authoritative DNS provider. This approach has the advantage that the information is secured by the CoT of the DNSSEC. In addition, the DNSSEC implicitly provides a key roll over mechanism where the Network Authority simply signs a new ZSK with its KSK if needed. However, DNS records are replaceable by domain owners without logging. Thus, this solution is neither transparent nor accountable. Additionally, as in OCSP a trade-off between propagation and dependency exists.

Key Server: In practice PGP uses key servers for distributing issued and revoked certificates. Those must be pulled by providing the name, email or fingerprint of the key. Therefore, as in OCSP a trade-off between propagation and dependency exists. Equally, key servers are not accountable or transparent and equivocate answers are possible.

Comment on Blockchain: Using the underlying blockchain system to distribute issued and revoked certificates onchain would provide an accountable and transparent solution. However, in contrast to writing Open Data a revoked certificate requires a low write latency. In this case, the performance of distributing a revoked certificate mainly depends on the performance of the consensus algorithm. However, as mentioned above it is assumed that the blockchain system uses a consensus algorithm, like PBFT, that provides strong consistency. According to chapter 2.5.5, such a consensus algorithm promises a high write latency especially with a growing number of nodes. Moreover, distributing issued and revoked certificates onchain may lead to poor modularity. Therefore, an onchain solution is disregarded in this use case. Moreover, a separate blockchain system could be used were the nodes are only allowed to read the blockchain and the Network Authority is the only entity allowed to write the blockchain. However, the use of an existing blockchain framework for the distribution of issued and revoked certificates would unnecessarily inflate the solution for this use case.

The comparison of distribution is summarized in table 3.2 where requirements are seen as equally important. All solutions are used in a pull-based fashion, since the clients are not known in the environment where these solutions are practically used. In addition, all solutions are classified as independent, since the pulling of certificate states can be performed on a regular basis. Therefore, the last known state of a certificate can be used. Using OCSP, key servers, or DNS to distribute information does not meet the requirements because it is neither accountable nor transparent. Lastly, using CT would miss the handling of revoked certificates but provides accountability and transparency. In conclusion, a custom design gives the freedom to meet the requirements. For this, a database is elaborated that is publicly available, verifiable and append-only, stores issued and revoked certificates and is replicated across all nodes. Finally, the design of this database will be inspired by blockchain technology and CT.

Requirement	**OCSP**	**CT**	**DNS/DNSSEC**	**Key Server**
Type	revoked certificates	issued certificates	both	both
Accountable	no	yes	no	no
Transparent	no	yes	no	no
Independence	yes	yes	yes	yes

Table 3.2: Comparison of the Distribution of Issued and Revoked Certificates

4 Design of the BODE-PKI

As concluded in chapter 3, a custom design of a dedicated BODE-PKI provides the freedom to meet the derived requirements. Based on the previous analysis, this chapter elaborates on this design. The main part of this design is an authorization system that manages the access control of the underlying blockchain system. Central part of this overview is how information is added to the system and how information is distributed across all participating nodes. For this overview, the system architecture is outlined, the authorization database and the replication of the authorization database are described, and finally the onboarding process and the revocation of a node is specified.

4.1 System Architecture

The designed BODE-PKI is based on a distributed system that is dedicated to manage authorization in terms of writing. This authorization system contains one authorization primary and multiple authorization replicas, each holding a copy of the authorization database. The authorization primary is operated by the Network Authority and is the only one authorized to write to the authorization database. Every update is propagated from the authorization primary to the authorization replicas. Each authorization replica is paired with one node of the blockchain system and is also operated by the respective Node Operator. The authorization replica works as a second-layer application that replicates the authorization database and updates the access control configuration of the associated blockchain node accordingly. Both the authorization replica and the blockchain node are assumed to run under the same location and are joining the system as a pair. Consequently, the blockchain node cannot participate in the blockchain system without receiving instructions from the associated authorization replica. To provide a better understanding, a schematic overview of the system architecture is depicted in figure 4.1.

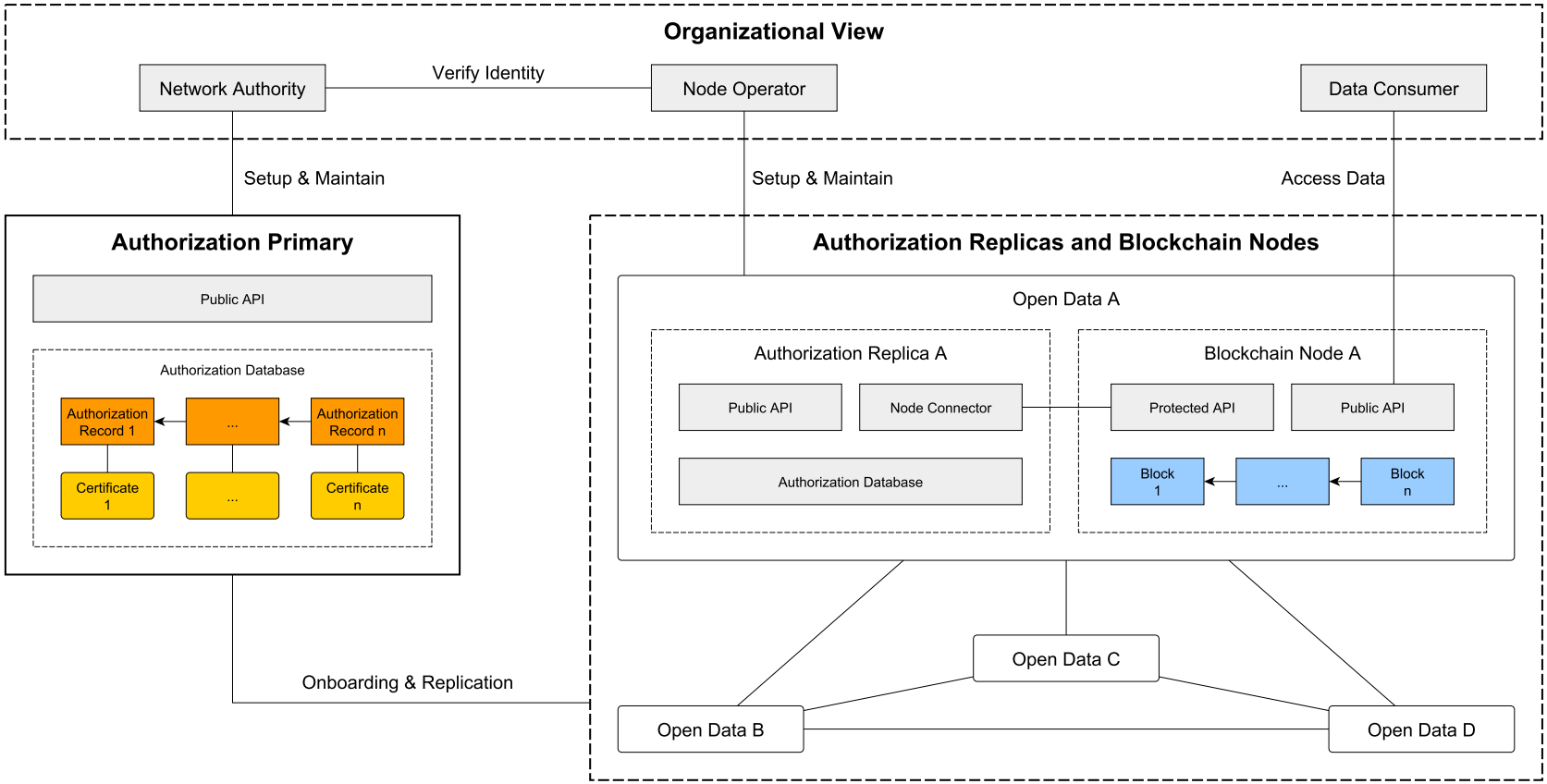

Figure 4.1: Architecture of the BODE-PKI

4.2 Authorization Database

Each record of the authorization database consist of one certificate with an Universal Unique Identifier (UUID), a public key and an endpoint. Any certificate that has been added to the authorization database counts as valid. Additionally, a value indicates whether the certificate is revoked or not. As described above, the authorization replica and the associated blockchain node are considered as one unit. Consequently, each authorization record always contains a certificate about one pair. The authorization database allows appending new records only in chronological order. Every new certificate or update to an existing certificate must be stored in a new record and appended to the database. In addition, the authorization database is publicly available and any inconsistent replication of the authorization database can be detected by comparing replicated copies. Thereby, the authorization system provides transparency and accountability through its publicly available, verifiable and append-only authorization database.

To ensure accountability and transparency, an authorization replica must be able to verify that the content and the order of a new version of the authorization database is consistent with the old version of the authorization database. Therefore, it needs to verify that old records are followed by new records and are still present in the same order. Additionally, a authorization replica must be able to verify authenticity of the content and the order of the new version. Inspired by blockchain technology and Certificate Transparency (CT), records are hashed and also contain the hash of the previous record. The latest hash is called the root hash and is signed by the authorization primary. Then, the content and the order can be verified by going through all records iteratively, verifying the chain of hashes and validating the signed root hash. Basically, this structure is an unbalanced merkle tree were the new root hash is determined by concatenating the old root hash with the hash of the new record. Unlike a balanced merkle tree, the number of hashes to be calculated is reduced when a new record is inserted, since the tree does not need to be kept in balance. The advantages of a balanced merkle tree, like the efficient audit proof in CT, are disregarded as they are not used in this design. If it becomes necessary to efficiently proof existence of a record or proof consistency of two database versions without replicating the authorization database, a balanced merkle tree might be the more sensible solution.

4.3 Replication

Each time a record is added to the authorization database, the authorization primary pushes this update to the authorization replicas. In order to reduce the amount of data that is transferred in the system, the authorization primary ensures that only new records are sent to the authorization replicas. To check which records are new for the authorization replica, the authorization primary fetches the current signed root hash of the authorization replica. Then the update message includes all new records according to the fetched signed root hash and the new signed root hash. To check that the update message is correct, the authorization replica at first verifies the new signed root hash and at second the consistency. For the latter, the authorization replica checks if its current root hash is included in the first new record, the new records show a valid chain of hashes and the new signed root hash belongs to the hash of the last new record. Then, all records are added to the database in the same order and the current signed root hash is set to the new signed root hash.

To speed up the propagation of an update, authorization replicas also push updates to other authorization replicas immediately after processing. Therefore, they proceed in the same way as the authorization primary. For the sake of completeness, if the fetched signed root hash is equal to the own view of the signed root hash no update has to be sent accordingly. Moreover, a authorization replica has to check that it is out of sync with the other authorization replicas. Inspired by CT, authorization replicas use their current view of the signed root hash to request an update from other authorization replicas or to verify that they have a consistent view of the authorization database. Therefore, authorization replicas exchange their view of the signed root hash and compare it with others. During the exchange with other authorization replicas, several cases may arise, where in the following the authorization replica that requests an update is denoted as A and the authorization replica that is requested is denoted as B:

1. B has the same view of the received signed root hash. Then, B answers with its current view of the signed root hash, so that A can verify that both see the same version of the authorization database.

2. B knows the received signed root hash but already has a newer version of the authorization database. Then, B answers with an update message, so that A can synchronize its authorization database.

3. *B* has never seen the received signed root hash before. Then, *B* answers with its current view of the signed root hash, so that *A* can verify that either *B* has an older version of the authorization database or both have a different view of the authorization database:

    a) *A* knows the received signed root hash but already has a newer version of the authorization database. Then, *A* answers with an update message, so that *B* can synchronize its authorization database.

    b) *A* has never seen the received signed root hash before. Then, both see an inconsistent version of the authorization database.

In case of 3b, the authorization primary is faulty and has distributed an inconsistent view of the authorization database. Then the authorization primary may have been compromised and the Network Authority must be informed immediately. To recover from such an attack, the authorization primary needs a new key pair and the public key has to be distributed across all authorization replicas. Additionally, all authorization replicas must go back to the last known state were all authorization replicas had a consistent view of the authorization database. Then the first update signed with the new private key of the authorization primary can indicate the last known consistent state. However, this is not the focus of this work and such a case may be treated differently in practice.

4.4 Onboarding

The onboarding process requires that the identity of a node has been verified through an out-of-band agreement between the corresponding Node Operator and the Network Authority. During identity verification, the Node Operator passes the nodes public key and endpoint to the Network Authority. At this point, both is added to an internal allow-map of the authorization primary without adding a record to the authorization database. Before the latter can happen, the authorization replica must proof control over its public key and endpoint. As mentioned in chapter 3.3, this is done to verify that both are technically usable and correct. Inspired by the ACME protocol, this proof of control can be processed automatically. At first, the authorization replica fetches a nonce from the authorization primary by providing its endpoint. This step is only allowed if the endpoint is present in the allow-map of the authorization primary. At second, the authorization replica signs the nonce with its private key and publishes the signature under a predefined location. At

third, the authorization replica triggers the authorization primary to verify the signature by providing the nonce. When triggered, the authorization primary looks up which endpoint belongs to the nonce, fetches the signature from the predefined location and verifies the signature with the public key which belongs to the endpoint. If all steps can be verified correctly, the authorization primary creates a new record containing a random UUID and the public key and endpoint of the corresponding node. In addition, the revocation status is initially set to *false*, and any changes can be recorded in subsequent records. Afterwards, the record is added to the authorization database and replicated across all authorization replicas. Then, all authorization replicas individually communicate with their associated blockchain node to update the access control configuration accordingly. Finally, the node is seen as authorized and can participate in the blockchain system.

4.5 Revocation

In addition to the onboarding process, the Network Authority can revoke write access from a node. When tasked with revocation, the authorization primary creates a new record that updates the current record that belongs to the node. For this, the authorization primary searches for the current record with the corresponding UUID provided by the Network Authority. An identical record is then created, except that the revocation status is set to *true*. Afterwards, the record is stored to the authorization database where the most recent version of the record holds the valid information. As before, immediately after storing, the new authorization database version is replicated across all authorization replicas. Whenever the Network Authority wishes to reverse the revocation, the same procedure is applied analogously by setting the revocation status back to *false*.

5 Implementation of the BODE-PKI

As described in chapter 4, the central element of the designed BODE-PKI is an authorization system. The authorization system consists of an authorization database managed by the authorization primary and replicated by the authorization replicas. Each authorization replica communicates with its associated blockchain node and updates the access control configuration according to the content of the authorization database. The designed BODE-PKI is implemented as a prototype and this chapter gives an overview of this implementation. The first part of this chapter will introduce implementation details with regard to the design specification of chapter 4. The second part of this chapter will go into detail about the connection between each authorization replica and the corresponding blockchain node. For this prototype, Hyperledger Besu is used to provide the underlying blockchain system. However, modularity is preserved by implementing an interchangeable connector that carries the exchange between each authorization replica and the corresponding blockchain node. Primarily, Hyperledger Besu serves as a proof of concept and was chosen because of its flexibility in a wide range of configuration parameters. If a different blockchain framework is required a suitable connector can be implemented. For this, the implementation provides a dedicated Java interface which specifies the necessary methods. In particular, the second part deals with the configuration of the Besu network and the implementation of the connector to the Besu node.

5.1 Authorization Primary and Replica

The implementation of the authorization primary and replica is written in Java and is based on the event-driven application framework Vert.x[1]. Both authorization primary and replica are wrapped up in one component which can be operated as either one. For communication in terms of onboarding and authorization database replication a dedicated API is provided. Depending on the type, a different set of methods is available. All methods of the API are listed and their functionality is described in appendix B. During

[1] Vert.x. (2021, January 21). https://vertx.io/

this section, their functionality is explained when mentioned. For interaction between the Network Authority and the authorization primary, and the Node Operator and the authorization replica, additional methods are provided in the aforesaid API. In practice, these methods would rather be provided via a dedicated Command Line Interface (CLI) or protected by a configurable API key. However, for evaluation purposes this manner was chosen for simplicity. In the following this section provides a technical overview of the onboarding process, the replication of the authorization database, the revocation of nodes and the node connector.

Onboarding

The onboarding process is depicted in figure 5.1. Basically, the onboarding process can be divided into two phases. For the first phase, the authorization primary holds an allow-map. Values are certificates and keys are nonces. Creating a certificate generates a corresponding UUID, which also serves as the nonce and as the key for the entry in the allow-map. The Network Authority can manage the allow-map through the *allow-map* API. To enable the onboarding process for a node, the Network Authority uses the *allow-map* API to add a new certificate to the allow-map. For this, the nodes public key and endpoint is required. The endpoint is split into host name, node port and authorization system port. Afterwards, a new certificate is created with a new random UUID and the input provided in the request. In addition, the revocation status is set to *false*. An example certificate as the result of the request is depicted in listing 5.1. The *allow-map* API can be used to list all certificates in the allow-map and review if something went wrong during the input. In this case, the *allow-map* API can be used to remove a certificate from the allow-map by providing the UUID of the corresponding certificate.

```
{
  "uuid": "2cbc132f-4faa-40d8-acbd-ac6bb3652d4b",
  "key": "16e14811[...]12b113aa",
  "host": "opendata.0.org",
  "asPort": 8080,
  "nodePort": 30303,
  "revoked": false
}
```

Listing 5.1: Example Certificate in the Allow-Map

For the second phase, the Node Operator initiates the onboarding process by using the *triggerOnboarding* API of the authorization replica. When triggered, the authorization replica requests a nonce by making a call to the *nonce* API of the authorization primary. For this, the endpoint of the authorization replica is required. The authorization primary goes through the allow-map and searches for the provided endpoint. If the authorization primary can find a matching certificate, the corresponding key of the allow-map is returned as the nonce to the authorization replica. The authorization replica signs this nonce and provides the signature under the *proof* API. After signing, the authorization replica triggers the authorization primary to verify the signed nonce by making a call to the *checkNonce* API. For this, the previously signed nonce is required. If the authorization primary knows the nonce, it checks the signature of the nonce by making a call to the *proof* API of the authorization replica. Then, the signature is returned and the authorization primary verifies the signature by using the public key that is contained in the corresponding certificate. If the signature is correct, the certificate is used to create a new record that is stored in the authorization database and the corresponding entry in the allow-map is removed. The new authorization database record consists of a hash, the hash of the predecessor record and the certificate. The actual hash is evolved by hashing the concatenation of the predecessor hash and the certificate hash. In this prototype SHA-256 is used for hashing purposes. An example of a record is depicted in listing 5.2. Finally, the new record is replicated across all authorization replicas and completing thereby the onboarding process.

```
{
  "hash": "2196546a[...]d1682466",
  "prevHash": "21a8a7e2[...]e106cbdd",
  "certificate": {
    "uuid": "2cbc132f-4faa-40d8-acbd-ac6bb3652d4b",
    "key": "16e14811[...]12b113aa",
    "host": "opendata.0.org",
    "asPort": 8080,
    "nodePort": 30303,
    "revoked": false
  }
}
```

Listing 5.2: Example Record in the Authorization Database

Replication

To keep all authorization replicas synchronized with the authorization primary, the mechanism described in chapter 4.3 is implemented. For retrieving the current view of the signed root hash, a call to the *signedRootHash* API is made. An update message is constructed by iterating the authorization database reversely until a record is found that matches with the retrieved signed root hash. Then all records that follow the matching one are considered new. The update message is sent by making a call to the *authorizationDatabase* API of the authorization replicas. The authorization primary performs replication immediately after appending a new record to the authorization database, and the authorization replicas perform replication immediately after processing an update. Synchronization between authorization replicas is done periodically with a configurable interval. If authorization replicas recognize an inconsistent view of the authorization database it is published under the *sync* API.

Revocation

The Network Authority can revoke a node by using the *revoke* API of the authorization primary. For this, the UUID of the certificate must be provided. The recent record with this UUID is retrieved and a new record is created that is equal, but the revoked field is reversed. Afterwards, the new record is stored in the authorization database and replicated across all authorization replicas.

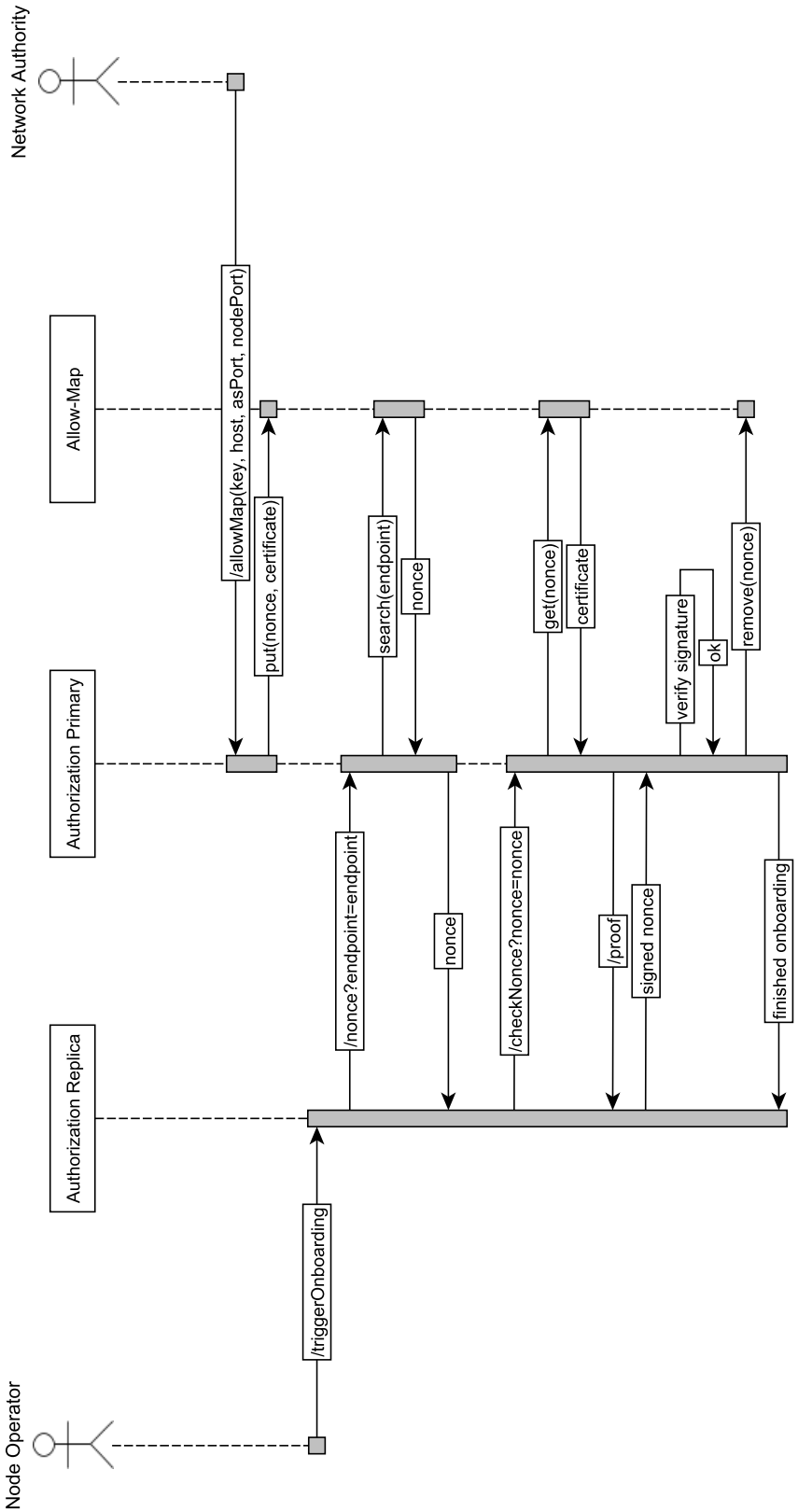

Figure 5.1: Sequence Diagram of the Onboarding Process

Node Connector

For the interaction between each authorization replica and the corresponding blockchain node, an interchangeable connector is provided, which must be implemented depending on the underlying blockchain system. The node connector implements the ability to update the nodes access control configuration depending on the authorization database. For that, a list with the most recent versions of all certificates is given to the node connector. The node connector processes this list and adjusts the nodes access control configuration accordingly. This process is done periodically for a configurable interval to keep the nodes configuration synchronized with the authorization database.

5.2 Besu Configuration and Connector

To meet the requirements of this prototype, the genesis file in listing 5.3 is used to configure the Besu network. For the consensus mechanism, IBFT 2.0 is selected in the *config* block. In order to create a low barrier, transactions must be free. Moreover, no incentive mechanism is needed as all participants are known and interested in the progress of the blockchain by nature. To do so, the *gasLimit* is set to the maximum value and the *blockreward* is set to zero in the *ibft2* block.

```
{
  "config": {
    "chainId": 2020,
    "muirglacierblock": 0,
    "contractSizeLimit": 2147483647,
    "ibft2": {
      "blockperiodseconds": 30,
      "epochlength": 30000,
      "requesttimeoutseconds": 60
    }
  },
  "nonce": "0x0",
  "timestamp": "0x58ee40ba",
  "gasLimit": "0x1fffffffffffff",
  "difficulty": "0x1",
  "mixHash": "0x63746963[...]6f6c6572616e6365",
  "coinbase": "0x0000000000000000000000000000000000000000",
  "alloc" : {},
  "extraData" : "0xf83ea000[...]f1808400000000c0"
}
```

Listing 5.3: Genesis File for the Besu Network Configuration

Moreover, listing 5.4 depicts the configuration for each Besu node. The *min-gas-price* is used to require no payment for consumed gas. In addition, nodes are configured to behave as a permissioned blockchain system. Data consumers should be able to read the blockchain without having access to any user credentials. Therefore, the prototype uses two communication channels as described in chapter 2.7. The APIs ETH, NET and TXPOOL are used over HTTP and the APIs ADMIN, PERM and IBFT are used over WebSocket[2]. Both the authorization replica and the Besu node must run on the same machine. The Node Operator must then ensure that the communication over WebSocket is protected and the communication over HTTP is open.

```
logging="INFO"
data-path="/opt/besu/data"
host-whitelist=["*"]

genesis-file="/opt/besu/config/genesis.json"
node-private-key-file="/opt/besu/keys/key"
min-gas-price=0

# rpc
rpc-http-enabled=true
rpc-http-host="0.0.0.0"
rpc-http-port=8545
rpc-http-cors-origins=["*"]
rpc-http-api=["ETH","NET","TXPOOL"]

# ws
rpc-ws-enabled=true
rpc-ws-host="0.0.0.0"
rpc-ws-port=8546
rpc-ws-api=["ADMIN","PERM","IBFT"]

# permissions
permissions-nodes-config-file-enabled=true
permissions-accounts-config-file-enabled=true

# txpool
tx-pool-retention-hours=1
```

Listing 5.4: Configuration File for each Besu Node

For the adjustment of the access control configuration, the Besu connector must process the current state of the authorization database. For this, each certificate is transformed to the enode url format and the corresponding account address. For further processing, a distinction is made between revoked and not revoked certificates. As a result, the

[2]In the evaluation, both communication channels are configured to allow all APIs, since some methods were used by the evaluation scripts and had to be accessible via HTTP.

Besu connector produces four lists: *nodesAllowed*, *nodesRevoked*, *accountsAllowed* and *accountsRevoked*. The lists *nodesAllowed* and *nodesRevoked* hold nodes in the enode url format and *accountsAllowed* and *accountsRevoked* hold the corresponding account addresses. Afterwards, the Besu connector uses the PERM API to retrieve the *nodes-allowlist* and *accounts-allowlist*. These are then compared with the prior constructed lists. For the comparison the following calculations are used:

- $nodesToAdd = nodesAllowed \setminus nodes\text{-}allowed$
- $nodesToRemove = nodesRevoked \cap nodes\text{-}allowed$
- $accountsToAdd = accountsAllowed \setminus nodes\text{-}allowed$
- $accountsToRemove = accountsRevoked \cap nodes\text{-}allowed$

If the comparison produces non empty lists, nodes and account addresses are added and removed accordingly through the PERM API. Moreover, the ADMIN API is used to connect to all nodes that are not revoked and to disconnect from all nodes that are revoked. Finally, the IBFT API is used to vote for all nodes that are not revoked and to vote against all nodes that are revoked to act as validators. A schematic overview of this procedure is shown in figure 5.2.

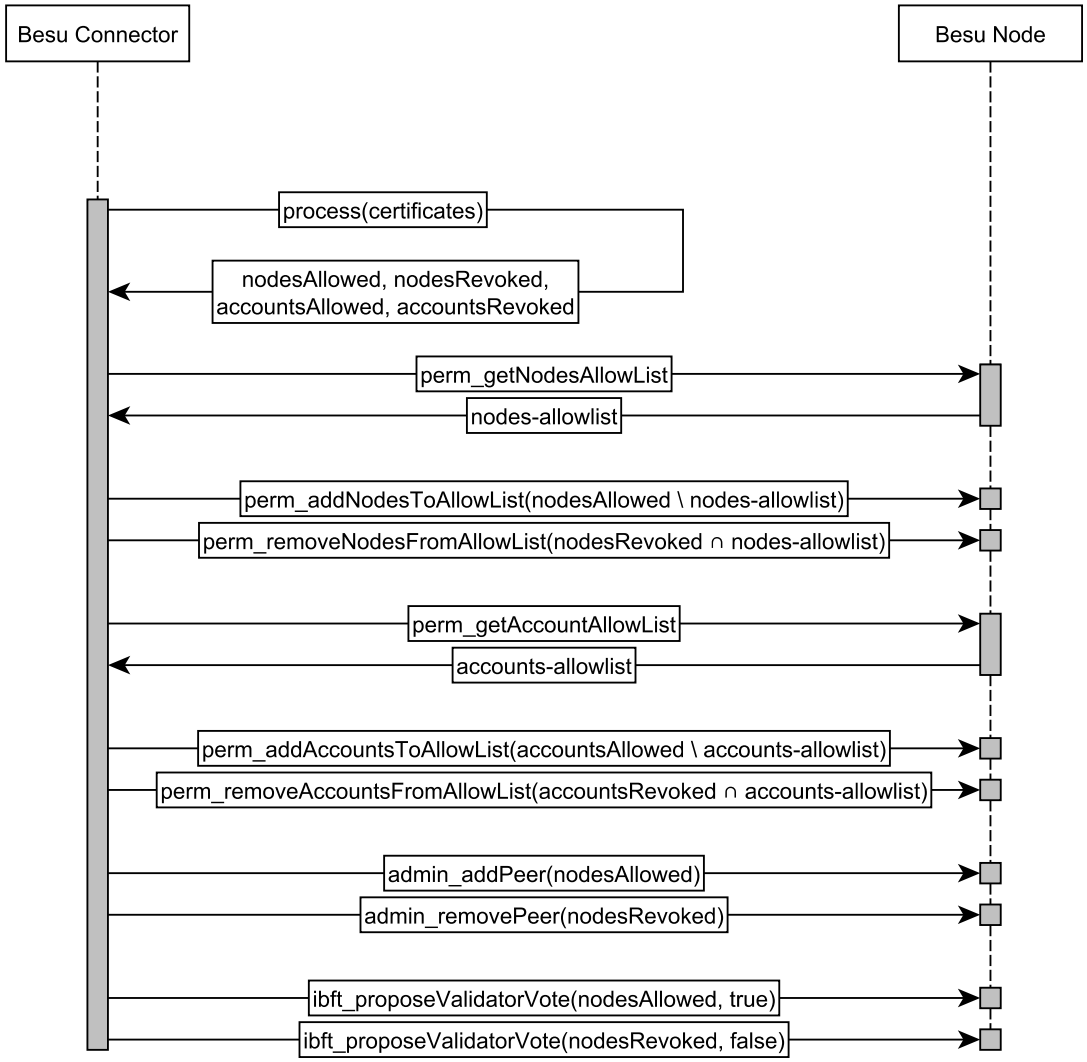

Figure 5.2: Sequence Diagram of the Access Control Adjustment

6 Evaluation

In this chapter the presented approach of a BODE-PKI is evaluated. In order to close the circle, the presented approach is evaluated against predefined requirements which were presented in chapter 3.3. In addition, tests on the prototype are carried out in order to demonstrate that the designed BODE-PKI can cope with a real-world environment by exhibit applicability and a performance that is defined by the requirements of a real-world Open Data ecosystem like the EDP. Appendix B provides instructions for reproducing and performing these tests with the implemented prototype.

6.1 Predefined Requirements

The compliance with some of the predefined requirements can only be verified with considerable effort, e.g., through user studies. In order to stick with the scope of this thesis, the compliance with these requirements is discussed in an argumentative manner. The compliance with other requirements will be verified in tests on the prototype. In the following, the compliance with the predefined requirements is evaluated in the same order as listed in chapter 3.3.

Identity Verification: In the designed BODE-PKI the Network Authority is considered as the crucial part of the identity verification process. The designed BODE-PKI leaves the verification of the identity to the Network Authority and does not describe any protocol. This gives the Network Authority the freedom to act according to its policy in this matter. Ultimately, the designed BODE-PKI simply offers an interface to add the result of the identity verification to allow the technical part of the onboarding. Therefore, the designed BODE-PKI meets the requirement of identity verification.

Proof of Control: In the designed BODE-PKI the onboarding process ensures that the provided public key and endpoint are technically usable and correct. Chapter 6.2.1 tests the onboarding process when testing the basic functionality.

Barrier to Join: In the designed BODE-PKI there are no costs in joining. In order to join an ecosystem, a Node Operator must have a technical background to some extent. This includes generating a desired key pair and starting a authorization replica and a blockchain node with the right configuration on a controlled machine. However, this seems to be in a feasible range and therefore the overall barrier to join is considered low.

Uniform Cryptosystem: In the designed BODE-PKI the public key which is used by a node in the underlying blockchain system is part of its certificate. In the onboarding process a predefined and uniform cryptosystem is used for signing the nonce. Thereby, the BODE-PKI ensures that all authorized nodes use a key pair from a uniform cryptosystem.

Independence: Both authorization replica and blockchain node have to communicate and run on the same machine. Beyond that, a node is able to operate independent from any elements of the BODE-PKI and external services. Therefore, the designed BODE-PKI meets the requirement of independence.

Transparency: In the designed BODE-PKI all certificates and their current state are available through the API of the authorization primary and of each authorization replica. As long as the authorization primary behaves correctly transparency is provided. Chapter 6.2.1 tests the transparency implicitly when testing the the basic functionality.

Accountability: In the designed BODE-PKI the underlying authorization database is designed as append-only and verifiable. Any update to the authorization database is signed by the authorization primary and distributed to the authorization replicas. If the primary behaves incorrectly by distributing inconsistent updates this behavior will be eventually determined by the authorization replicas. For this, the authorization replicas exchange their current view of the authorization database and publish inconsistent replication under the *sync* API. Chapter 6.2.1 tests the detection of inconsistent database replication by the authorization primary.

Modularity: In the designed BODE-PKI the authorization replica and the blockchain node are considered separately. The authorization replica adjusts the access control configuration of the associated blockchain node but the way of communication is designed modular. This is also reflected in the prototype where the implementation of a connector is required depending on the underlying blockchain system. Therefore, the designed BODE-PKI provides the desired modularity.

6.2 Applicability and Performance

A real-world simulation is constructed to evaluate the applicability and performance of the designed BODE-PKI. As of December 2020, the 10 biggest data publishers of the EDP provide more than 80% of the datasets due to the inequality in contribution. It is therefore sufficient for this simulation to deploy 1 authorization primary and 10 authorization replicas as well as 10 corresponding Besu nodes. For the test infrastructure all entities are deployed in a separate Docker[1] container and are running in a virtual network on a single machine. A network delay is simulated by using the traffic control utility[2]. More precisely, each outgoing packet with its destination to another container is delayed with a random amount sampled from a normal distribution with a mean of 1000*ms* and a jitter of 500*ms*. Finally, the EDP provides the sufficient and necessary test data in form of metadata. These datasets are accessible through a public API provided by the EDP[3]. For this evaluation, it is sufficient to store a dataset in the *input data* field of a transaction and set the account address of the data publisher both to the *from* and to the *to* field.

The following sections include a basic functionality test, a synchronization test and an access control test. The basic functionality test also implicitly checks, as mentioned in section 6.1, whether the designed BODE-PKI meets the proof of control requirement and the requirements of transparency and accountability. Each section is viewed as independently and before each test the overall system will be reset to the initial state where the authorization database and the blockchain is empty. Each chapter will provide a description of what is tested and what is evaluated, the results of the test and the assessment of the results. For simplicity, all authorization replicas and their corresponding blockchain node are referred to as node-replica pairs. In addition, blockchain nodes, authorization replicas, and node-replica pairs are enumerated from 0 to 9 inclusive.

6.2.1 Basic Functionality

The basic functionality test evaluates the applicability of the designed BODE-PKI to some extent. In addition, the proof of control requirement is evaluated by testing the onboarding and the compliance of transparency is evaluated by updating the authorization database and verifying its content. Moreover, the accountability requirement is tested by

[1] Docker. (2021, January 21). https://www.docker.com/
[2] Traffic Control. (2021, January 21). https://linux.die.net/man/8/tc
[3] EDP Hub Search. (2021, January 21). https://www.ppe-aws.europeandataportal.eu/data/search/

constructing a scenario where the authorization primary replicates an inconsistent view of the authorization database.

At first, the onboarding is processed for all 10 node-replica pairs. For this, each onboarding is processed step by step. At first, a node-replica pair is added to the allow-map of the authorization primary. Then the occurrence of the node-replica pair is reviewed in the allow-map. Afterwards, the onboarding is triggered. These steps are repeated for all node-replica pairs. Finally, it is verified that for all node-replica pairs a belonging certificate is present in the authorization database of the authorization primary. When all 10 node-replica pairs are on board, the consistency is verified by comparing the authorization database of the authorization primary with each of the authorization replicas. If all have a consistent view of the authorization database, the Besu configuration is reviewed. For this, the access control configuration is compared by using the PERM API, where blockchain nodes should return the same *nodes-allowlist* and *acconts-allowlist*. Moreover, the connectivity to other blockchain nodes is inspected and compared by using the NET API, where blockchain nodes should return the number 9. Afterwards, the basic access control is tested. For this, 10 sets of transactions are created. Each set consists of 10 transactions, each signed by a different blockchain node. Then each set is sent to a different blockchain node of the ecosystem. Afterwards, it is expected that these previously sent transactions can be found in the blockchain. In the end, one node-replica pair is revoked. Then, a transaction is created and signed by the associated blockchain node which was revoked before. This transaction is sent to all 10 blockchain nodes and a rejection is expected.

Finally, the accountability is tested. For this, the ecosystem is reset to its initial state. Afterwards, node-replica pair 0 and 1 join the system and the authorization database of the authorization primary is mirrored. Then, node-replica pair 1 is stopped and afterwards node-replica pair 2 joins. Now, node-replica pair 1 does not know about the onboarding of node-replica pair 2. Afterwards, node-replica pair 0, 2 and the authorization primary are stopped. Then, the previously mirrored authorization database of the authorization primary is restored and the authorization primary is restarted. Now, the authorization database does not consist information about node-replica pair 2 that joined before. Afterwards, node-replica pair 1 is restarted and node-replica pair 3 joins. Then, node-replica pair 0 and 2 are restarted. After exchanging information during synchronization, both node-replica pair 0 and 1 should recognize that the authorization primary has replicated an inconsistent view of the authorization database and should publish it under the *sync* API.

Table 6.1 summarizes the phases of the basic functionality test and describes the respective expectations. For each phase it was marked whether the expectation was met or not. Finally, all phases of the basic functionality test were passed as expected and the applicability of the designed BODE-PKI was shown to some extent. Moreover, the basic functionality test shows that the designed BODE-PKI meets the proof of control requirement and complies with the requirements of transparency and accountability.

Phase	**Expectation**	**Passed**
Allowance	After adding a certificate for a node-replica pair to the allow-map, it can be verified that this certificate occurs in the allow-map.	✓
Onboarding	After processing the onboarding of an allowed node-replica pair, it can be verified that the belonging certificate occurs in the authorization database of the authorization primary.	✓
Synchronization	After adding all node-replica pairs, it can be verified that the authorization primary and each authorization replica have eventually the same view of the authorization database.	✓
Perm. Configuration	After adding all node-replica pairs, it can be verified that each blockchain node is eventually configured with the same *nodes-allowlist* and *accounts-allowlist*.	✓
Connectivity	After adding all node-replica pairs, it can be verified that each blockchain node is eventually connected to the same number of other blockchain nodes.	✓
Access Control I.	After adding all node-replica pairs, it can be verified that transactions that are signed by blockchain nodes which are part of the ecosystem are accepted and can be found in the blockchain.	✓
Access Control II.	After adding all node-replica pairs and revoking one of them, it can be verified that transactions that are signed by the revoked blockchain node are rejected and can not be found in the blockchain.	✓
Accountability	After the authorization primary replicates an inconsistent version of the authorization database, eventually authorization replicas detect this behavior and publish it under the *sync* API.	✓

Table 6.1: Results of the Basic Functionality Test

6.2.2 Synchronization

The synchronization test evaluates the applicability of the designed BODE-PKI in more detail. In order to exhibit applicability, it is essential that the authorization primary and all authorization replicas eventually have a consistent view of the authorization database with regard to crashes and updates of the authorization database. To simulate a crash, the node-replica pair is stopped at some point and restarted at some later point. For the BODE-PKI, an update of the authorization database occurs whenever a new node-replica pair is joining, a node-replica pair is revoked or by turning back a revocation. In the first phase, the onboarding process is processed for all node-replica pairs. In the meantime, node-replica pairs that are already part of the ecosystem are artificially crashed randomly. During the entire phase, the synchronization of each authorization replica is recorded. For this, the authorization database of the authorization primary and each authorization replica is requested at each time step. Afterwards, the number of authorization replicas providing the same authorization database as the authorization primary is counted. At some point, all authorization replicas recover from crashes and it is expected that all authorization replicas are eventually synchronized with the authorization primary. In the second phase, all node-replica pairs are revoked one after another and then the revocation is pulled back step by step. As in the first phase, there are crashes simulated and the synchronization of each authorization replica is recorded.

Figure 6.1[4] shows the number of synchronized authorization replicas over time while onboarding all node-replica pairs in the first phase. It can be observed that the number of synchronized authorization replicas increases from 0 to 10 over time. In between there are several jumps to 0 synchronized authorization replicas. Shortly after these jumps, the number of synchronized authorization replicas jumps to a higher value. These jumps can be explained by the combination of a authorization database update and the network delay, since only the authorization primary knows about the update for a short time interval. Smaller jumps, on the other hand, are mainly caused by crashes. Finally, the prototype reaches the desired number of synchronized authorization replicas.

[4]For further insight, figure A.1 as part of the appendix shows the individual synchronization for each authorization replica and when they were not reachable due to crashes.

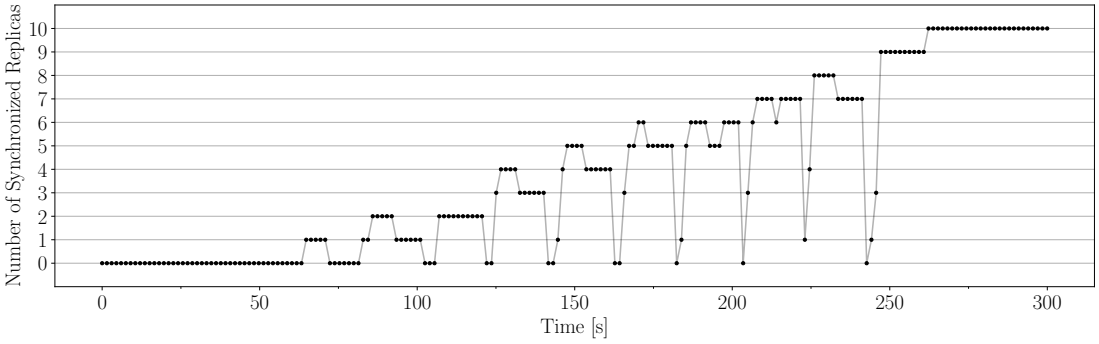

Figure 6.1: Synchronization over Time during Onboarding with Crashes

Figure 6.2[5] shows the number of synchronized authorization replicas over time while all node-replica pairs are incrementally revoked and then reactivated. It can be observed that the number of synchronized authorization replicas decreases from 10 to 0 and then increases from 0 to 10 over time. As in figure 6.1 there are small jumps due to crashes and big jumps due to authorization database updates. In the end, the prototype reaches the desired number of synchronized replicas.

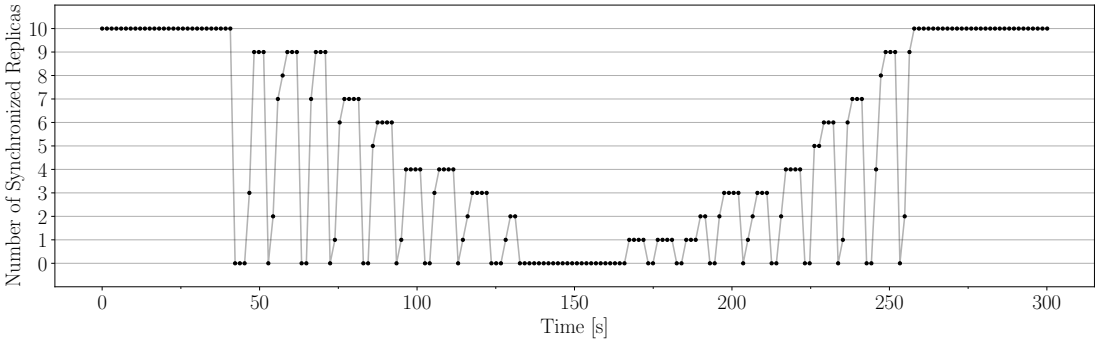

Figure 6.2: Synchronization over Time during Revocation with Crashes

Finally, it can be assessed that the prototype shows the desired behavior and that it exhibits applicability in terms of synchronization.

[5]For further insight, figure A.2 as part of the appendix shows the individual synchronization for each authorization replica and when they were not reachable due to crashes.

6.2.3 Access Control

The access control test evaluates the performance of the designed BODE-PKI in a simulated real-world environment. In each phase, transactions are created and signed with the private key of blockchain node 9. Transactions are sent to randomly chosen nodes with a constant transaction rate of one transaction per second. During each phase, there occurs an update of the authorization database related to blockchain node 9. To measure performance, the number of transactions that were incorrectly written or rather incorrectly not written to the blockchain is counted. In addition, the delay with which transactions are again correctly written or rather not written to the blockchain is measured. For the test, initially, the node-replica pairs from 0 to 8 are on board and synchronized. In the first phase the node-replica pair 9 joins, in the second phase it is revoked and in the third phase it is reactivated. In the last phase the node-replica pair 9 is revoked and then reactivated. In each phase, transactions are sent at an interval of 100s before, after and between these updates of the authorization database. After each phase, it is ensured that the transaction pool of all blockchain nodes is empty. If this is given, the next phase follows. In the end, when all phases are completed, a verification is made for each transaction whether it was written to the blockchain or not. Then the state of the node-replica pair 9 at the respective point in time defines the expectation in this regard.

Figures 6.3-6.6 show the results of the aforementioned four phases. In all figures, transactions are shown according to the point in time they were issued and the blockchain node they were sent to. In addition, a transaction is marked green if it was written to the blockchain and marked red otherwise. Both the time of the authorization database update and blockchain node 9 as the signer of transactions have been highlighted in bold.

In the first phase, shown in figure 6.3, the onboarding was triggered for node-replica pair 9 at $t_{onboarding} = 100s$. Viewing at the data from $t_{onboarding}$, it can be observed that 18 transactions were rejected due to a delay that occurs during replication of the authorization database.

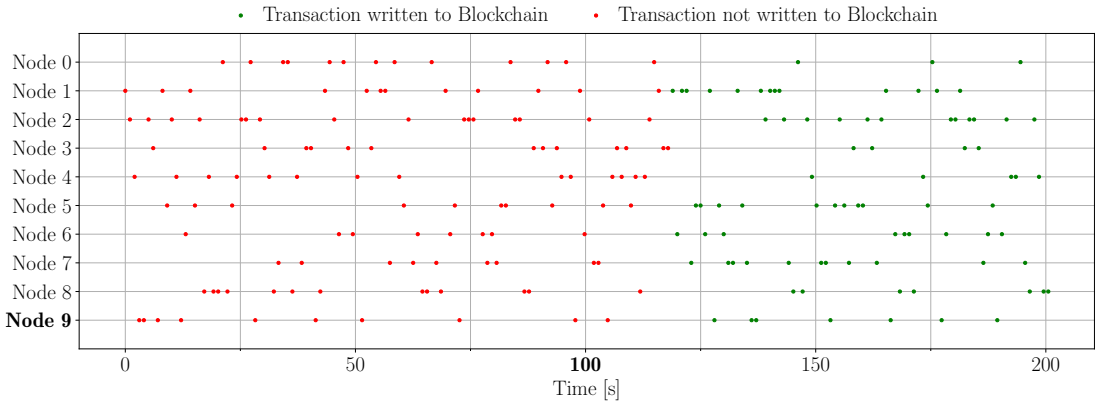

Figure 6.3: Access Control during Onboarding

In the second phase, shown in figure 6.4, the node-replica pair 9 was revoked at $t_{revocation} = 100s$. Apparently, 6 transactions were written incorrectly to the blockchain even though they were issued after $t_{revocation}$. Those transactions found there way into the transaction pool and were distributed before all node-replica pairs knew about the revocation.

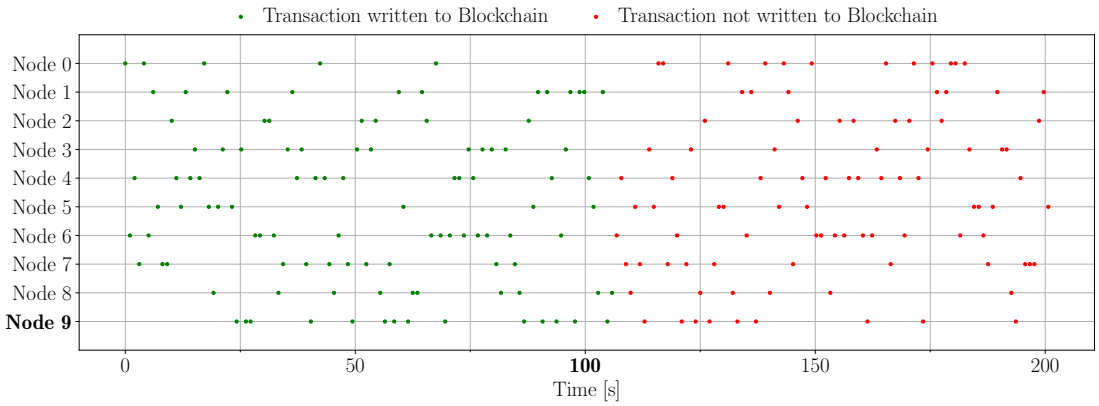

Figure 6.4: Access Control during Revocation

In the third phase, shown in figure 6.5, the node-replica pair 9 was reactivated at $t_{reactivation} = 100s$. Viewing at the data from $t_{reactivation}$, it can be observed that 12 transactions were rejected due to a delay that occurs during replication of the authorization database.

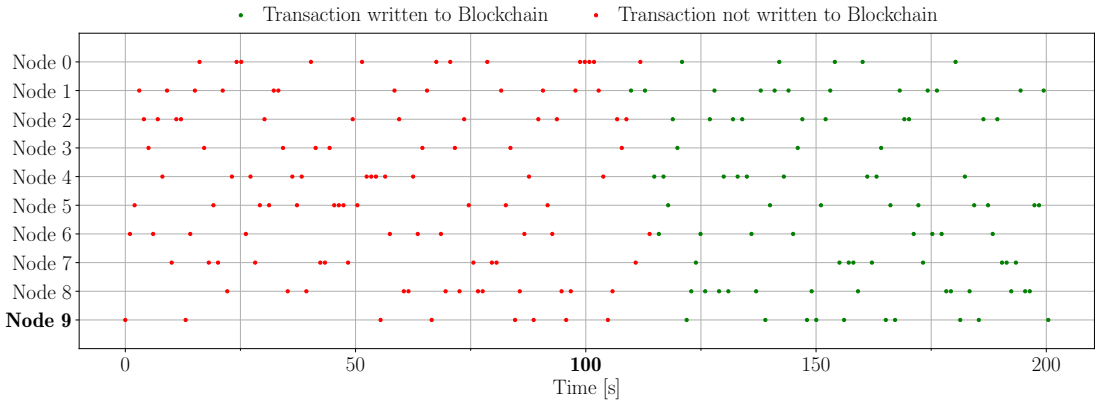

Figure 6.5: Access Control during Reactivation

In the fourth phase, shown in figure 6.6, the node-replica pair 9 was revoked at $t_{revocation} = 100s$ and reactivated at $t_{reactivation} = 200s$. Apparently, 13 transactions were written to the blockchain even though they were issued after $t_{revocation}$ and before $t_{reactivation}$. In contrast to the second phase, there is a slight increase, which can be explained by the fact that transactions which made it into the transaction pool are later valid again after the reactivation. For the reactivation a similar behavior compared to the third phase can be observed. Viewing at the data from $t_{reactivation}$, it can be observed that 13 transactions were rejected due to a delay that occurs during replication of the authorization database.

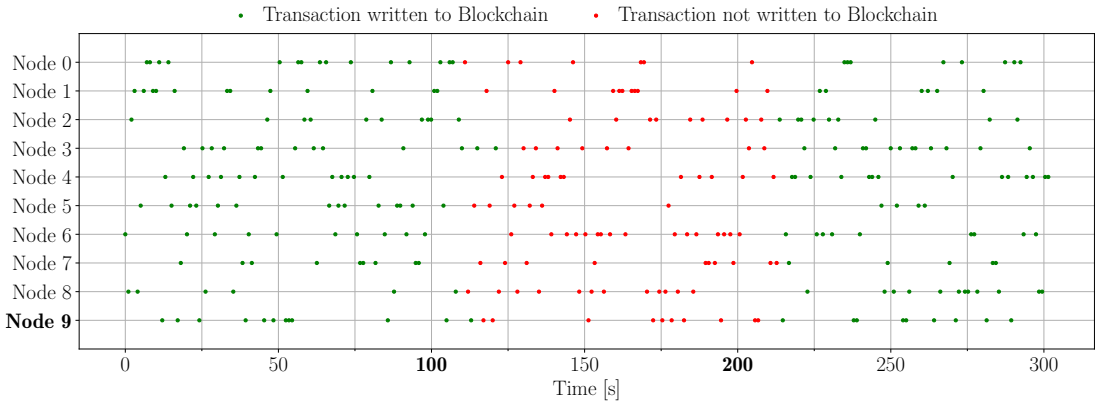

Figure 6.6: Access Control during Revocation and Reactivation

Table 6.2 summarizes for each phase the number of incorrect transactions, the delay of the first successful transaction, and the last incorrect transaction after the authorization database has been updated.

Phase	Number of incorrect transactions	Delay of first successful transaction [s]	Delay of last incorrect transaction [s]
1. Onboarding	18	18.93	17.92
2. Revocation	6	6.83	5.83
3. Reactivation	12	9.82	13.85
4. Revocation	13	10.93	20.98
4. Reactivation	13	13.71	12.70

Table 6.2: Results of the Access Control Test

To assess those results, one should distinguish between phases were the node-replica pair joins the ecosystem through onboarding or reactivation and phases were the node-replica pair leaves the ecosystem through revocation. In case of joining, the number of incorrect transactions is low. Moreover, the Node Operator has the possibility to reissue failed transactions so that the number even becomes more insignificant. In addition, the delay is considered to be within a adequate range as Node Operators normally would not issue transactions right after the onboarding or the reactivation takes place. In case of leaving, the second test phase shows also a low number of incorrect transactions. A blockchain node that behaves incorrectly could issue transactions at a higher frequency, resulting in a higher number of incorrect transactions. However, the delay with which the system responds seems within an acceptable range. Compared to the second phase, the number of incorrect transactions is slightly higher in the fourth phase after revocation. However, those transactions become valid again as soon as the reactivation is processed. It should be noted that there exists no real use case for the fourth phase, except for the accidental revocation with a subsequent reactivation. However, this case should be prevented by the Network Authority. Finally, the prototype shows performance in terms of access control within a reasonable range.

7 Conclusion and Outlook

Several studies have addressed the symbiosis of Open Data and blockchain. The application of blockchain to support Open Data ecosystems promises many benefits, such as no single point of failure, increased availability of metadata, a virtual central access point under decentralized control, increased interoperability and more control over the quality of metadata. Open Data is readable by everyone and only known data providers and data publishers have the right to write. Open Data ecosystems need a convenient process for onboarding new data providers and data publishers, and based on that, they need a practicable process for managing participants in the ecosystem. The same applies to a blockchain-based Open Data ecosystem. To reflect the access control of Open Data, such an ecosystem can be based on a public permissioned blockchain system. For this, data publishers and data providers must have a certificate that links the identity to a verified key pair to write to the blockchain and a verified endpoint to operate a blockchain node. So far, current research has not addressed how a new node can join a blockchain-based Open Data ecosystem while taking the existing governance and authentication structures of Open Data into account. We started with the hypothesis that a dedicated Public Key Infrastructure for a Blockchain-based Open Data Ecosystem (BODE-PKI) can handle the certification of new nodes and the distribution of certificates and their revocation status across all participating nodes. We have derived requirements for such a BODE-PKI and analyzed existing PKIs to see if they can meet these requirements. We concluded that a custom design provides the desired freedom in this regard. From here, we designed a BODE-PKI that is based on an authorization system that is inspired by blockchain technology and Certificate Transparency (CT). This authorization system is based on a publicly, verifiable and append-only authorization database that stores certificates and their revocation status. In addition, the authorization database is replicated across the ecosystem and thereby manages the access control of the underlying blockchain system. An authority, which we call the Network Authority, manages the participants of the ecosystem and has full control over the authorization database. For the node onboarding, we designed a dedicated process that is inspired by the ACME protocol. This requires out-of-band communication between the Network Authority and the operator of the node,

which we call the Node Operator, to verify identity and to exchange the corresponding public key and endpoint. Finally, we evaluated the designed BODE-PKI against the predefined requirements. Partly through basic functionality tests and partly through functional analysis, we have shown that the requirements are satisfied. Moreover, we evaluated the designed BODE-PKI for its applicability through basic functionality and synchronization tests and for its performance through access control tests. Our evaluation has shown that the designed BODE-PKI demonstrates applicability in terms of its basic functionality and synchronization of the authorization database across the ecosystem. In addition, we have shown that the authorization system responds to authorization database updates in a reasonable time range, providing the desired performance for a production application.

The presented onboarding process requires human interaction in three phases. First, the Network Authority has to verify the identity of the Node Operator, second, the Network Authority has to enter the information into the authorization system, and third, the Node Operator has to trigger the onboarding process. Beyond that, everything else is automated and requires no further human interaction. Importantly, no configuration file needs to be adjusted or any service must be restarted. The underlying blockchain system can proceed whenever the authorization database is updated.

For further development, human interaction could be kept to the out-of-band identity verification only. An additional application could be used to enter information by the Node Operator and generate an associated QR code. The Network Authority would then trigger the onboarding process directly by scanning the QR code. In addition, digital identities will improve in the future. They become usable and may also contain a public key that is usable in the blockchain system. This would eliminate the need for identity verification and allow for the best possible automation.

Open Data is not a domain that motivates attackers. Thus, a centralized Network Authority can be considered sufficiently secure. Nevertheless, in practice, precautions should be taken in the event of an attack. If the private key of the Network Authority falls into wrong hands, a suitable procedure must exist to establish a new key pair in the ecosystem. As mentioned in the analysis, the DNSSEC can be used to implement a suitable key rollover mechanism. Finally, Open Data ecosystems such as the EDP have a hierarchical structure. In practice, delegation of the onboarding process could be important. Since the presented approach omits this, a suitable method can be designed in this regard.

A Figures

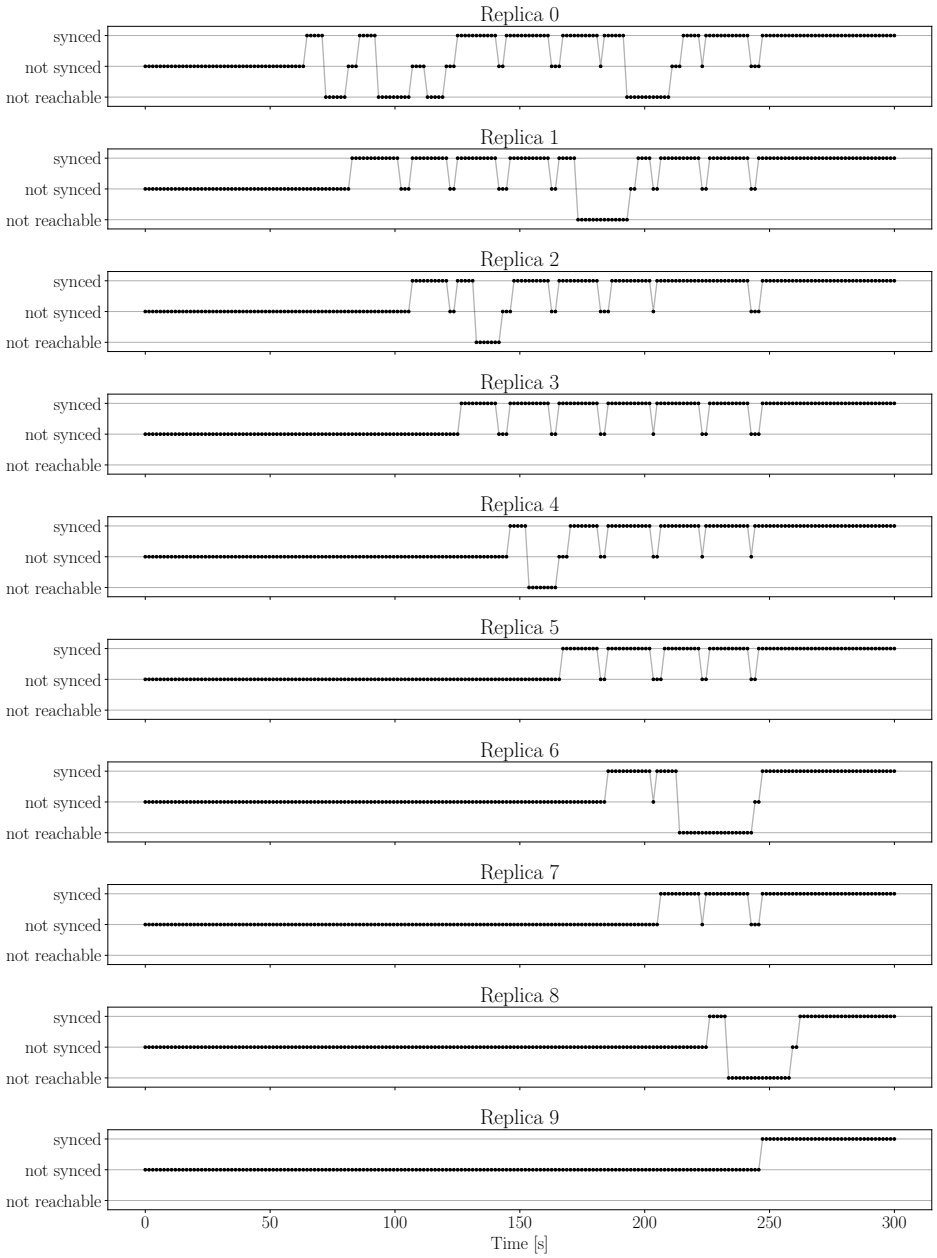

Figure A.1: Individual Synchronization over Time during Onboarding with Crashes

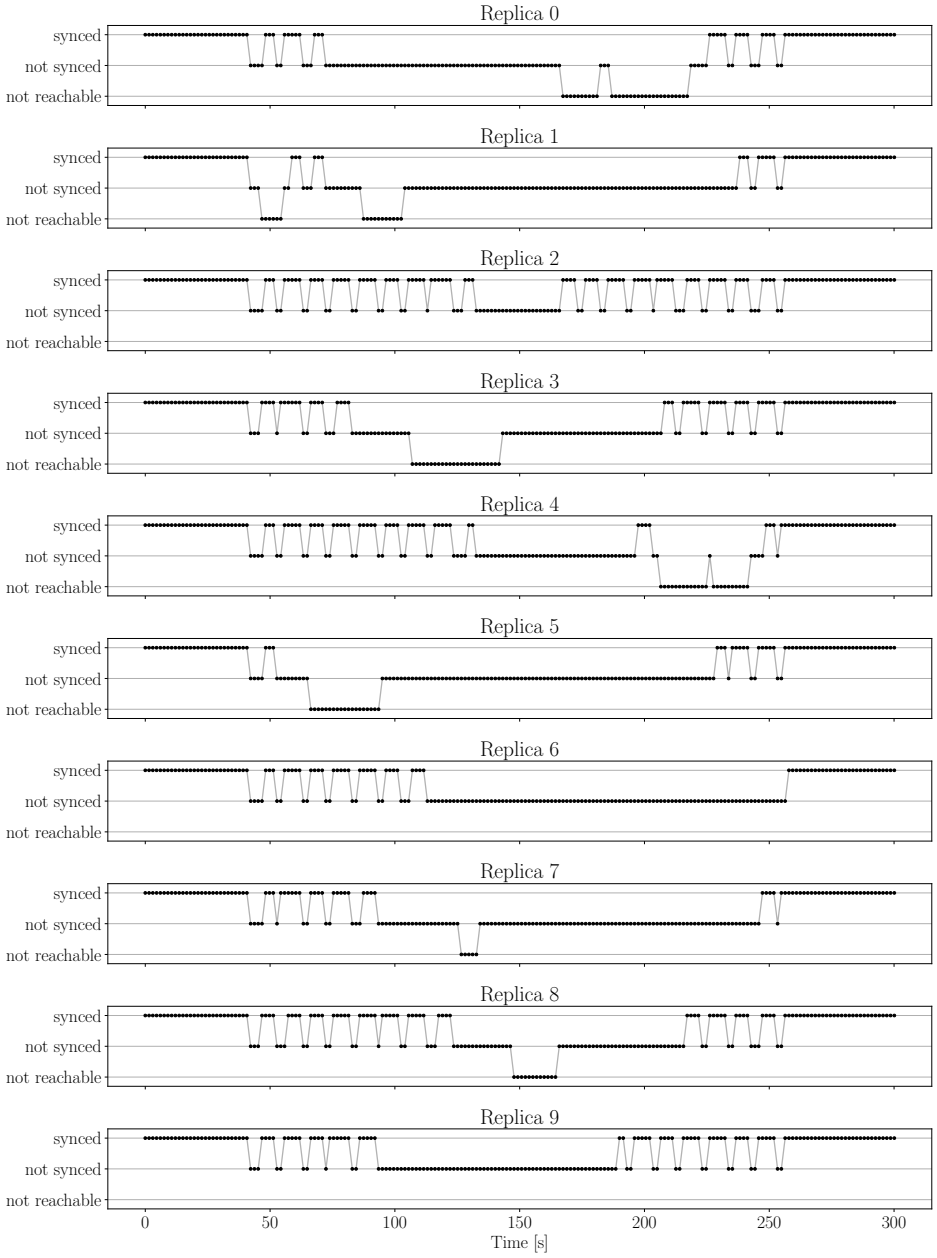

Figure A.2: Individual Synchronization over Time during Revocation with Crashes

B Prototype

This chapter provides instructions on how to perform the tests with the prototype. These tests were used for the evaluation in chapter 6. This chapter includes the prerequisites, a description of the system under test, the folder structure, how the prototype is build and the actual execution of the tests. Moreover, a description of the API is provided.

Instructions

You can find the source code of this prototype either in the attachment of this thesis or in the GitHub repository under the following location:

```
https://github.com/aaltenbernd/bode-pki
```

Prerequisites

In the following prerequisites are listed that are necessary to build the prototype and perform the tests:

- docker 20.10.2
- docker-compose 1.27.4
- maven 3.6.1
- java 11.0.6
- python 3.9.0
- pip 20.3.1

In addition, the following Python libraries are mandatory to run the tests: web3, numpy, grequests, matplotlib, pathlib. The following command installs the required libraries:

```
$ pip install web3 numpy grequests matplotlib pathlib
```

Test System

The specification of the system used to perform the tests are listed in the following:

- CPU: Intel Core i5-8265U CPU @ 1.60GHz 1.80 GHz, 4 Cores, 8 Threads
- Memory: 16 GB
- Operating System: Windows 10

Moreover, docker desktop was configured with 4 CPUs, 8 GB memory and 2 GB swap. You can configure the virtual resources under *Settings* | *Resources*. For the execution both Windows PowerShell and PyCharm/IntelliJ Terminal have been tested and are recommended. In contrast, Git Bash does not print any steps and is therefore not recommended.

Folder Structure

The following folder structure is present in the source code:

- bode-pki
  - bode-pki-authorization-system
    - conf
    - src
  - bode-pki-eval
    - docker
    - eval
    - results

Build

To build the prototype, please perform the following five steps:

1. Go to the evaluation folder:

```
$ cd ./bode-pki-eval/
```

2. Go to the build folder:

```
$ cd ./docker/build/
```

3. Make the build script executable:

```
$ chmod +x build.sh
```

4. Run the build script:

```
$ ./build.sh
```

5. Return to the evaluation folder:

```
$ cd ../../
```

Run

The tests are written Python and Python 3 is mandatory. If your Python 3 executable is reachable by 'python3' please replace it in the following commands accordingly. The main script is using a reset script. Both the main script and the reset script are located in the *eval* folder. The reset script is used to restore all docker container to their initial state between tests. If your docker is running as root under Unix you may have to run these tests with sudo.

Run all tests by:

```
$ python eval/main.py all
```

Run the basic functionality test by:

```
$ python eval/main.py basic
```

Run the synchronization test by:

```
$ python eval/main.py sync
```

Run the access control test by:

```
$ python eval/main.py access
```

After performing the tests, you will find the results in the results folder. The folder with the highest number contains the results of the last test run.

API

[GET] /allowMap - authorization primary
List all certificates in the allow-map.

[POST] /allowMap - authorization primary
Add a public key and an endpoint as a certificate to the allow-map.

[DELETE] /allowMap?uuid={uuid} - authorization primary
Remove a certificate from the allow-map.

[GET] /nonce?endpoint={endpoint} - authorization primary
Request a nonce for the proof of control by providing an endpoint. If the endpoint is known a nonce is returned.

[GET] /checkNonce?nonce={nonce} - authorization primary
Request a verification for the proof of control by providing a nonce. If the nonce is known and the signed nonce can be verified a positive answer is returned.

[GET] /proof - authorization replica
Request a signed nonce.

[GET] /triggerOnboarding - authorization replica
Trigger the onboarding process.

[GET] /revoke?uuid={uuid} - authorization primary
Create a new record with inverted *revoked* field by providing the UUID.

[GET] /certificates - authorization primary and replica
Request the state of the authorization database. A list of all certificates is returned but only the recent state of each certificate is contained.

[GET] /signedRootHash - authorization primary and replica
Request the signed root hash.

[GET] /authorizationDatabase - authorization primary and replica
Request the authorization database. The signed root hash and the authorization records are returned.

[PUT] /authorizationDatabase - authorization replica
Update the authorization database by providing the new signed root hash and new records. If the chain of hashes and the signed root hash is valid a positive answer is returned.

[GET] /sync - authorization replica
Request the current replication state. All known replicas and their state about synchronization are listed.

[PUT] /sync - authorization replica
The replicas use this endpoint for synchronization purposes internally. They use their view of the signed root hash as the content of this request.

Bibliography

[1] Kieron O'Hara. "Transparency, open data and trust in government: shaping the infosphere". In: *Proceedings of the 4th annual ACM web science conference*. 2012, pp. 223–232.

[2] Mauricio Solar, Luis Meijueiro, and Fernando Daniels. "A guide to implement open data in public agencies". In: *International conference on electronic government*. Springer. 2013, pp. 75–86.

[3] Katrin Braunschweig, Julian Eberius, Maik Thiele, and Wolfgang Lehner. "The state of open data". In: *Limits of current open data platforms* (2012).

[4] Fabian Kirstein, Kyriakos Stefanidis, Benjamin Dittwald, Simon Dutkowski, Sebastian Urbanek, and Manfred Hauswirth. "Piveau: A Large-Scale Open Data Management Platform Based on Semantic Web Technologies". In: *European Semantic Web Conference*. Springer. 2020, pp. 648–664.

[5] Esther Huyer and Gianfranco Cecconi. "Analytical Report 11: Re-use of PSI in the public sector". In: *European Commission* (2019).

[6] Anneke Zuiderwijk, Natalie Helbig, J Ramón Gil-García, and Marijn Janssen. "Special issue on innovation through open data: Guest editors' introduction". In: *Journal of theoretical and applied electronic commerce research* 9.2 (2014), pp. i–xiii.

[7] European Data Portal. *Open Data and Blockchain: a match made in heaven?* 2018. URL: `https://www.europeandataportal.eu/en/highlights/open-data-and-blockchain-match-made-heaven` (visited on 01/21/2021).

[8] Xiwei Xu, Ingo Weber, and Mark Staples. *Architecture for blockchain applications*. Springer, 2019.

[9] An Binh Tran, Xiwei Xu, Ingo Weber, Mark Staples, and Paul Rimba. "Regerator: a Registry Generator for Blockchain." In: *CAiSE-Forum-DC*. 2017, pp. 81–88.

[10] Elena García-Barriocanal, Salvador Sánchez-Alonso, and Miguel-Angel Sicilia. "Deploying metadata on blockchain technologies". In: *Research Conference on Metadata and Semantics Research*. Springer. 2017, pp. 38–49.

[11] Matthew English, Sören Auer, and John Domingue. "Block chain technologies & the semantic web: A framework for symbiotic development". In: *Computer Science Conference for University of Bonn Students, J. Lehmann, H. Thakkar, L. Halilaj, and R. Asmat, Eds*. sn. 2016, pp. 47–61.

[12] Allan Third and John Domingue. "LinkChains: Exploring the space of decentralised trustworthy Linked Data". In: (2017).

[13] Mirek Sopek, Przemyslaw Gradzki, Witold Kosowski, Dominik Kuziski, Rafa Trójczak, and Robert Trypuz. "GraphChain: a distributed database with explicit semantics and chained RDF graphs". In: *Companion Proceedings of the The Web Conference 2018*. 2018, pp. 1171–1178.

[14] Anh Le-Tuan, Darshan Hingu, Manfred Hauswirth, and Danh Le-Phuoc. "Incorporating blockchain into rdf store at the lightweight edge devices". In: *International Conference on Semantic Systems*. Springer, Cham. 2019, pp. 369–375.

[15] Russell L Ackoff. "From data to wisdom". In: *Journal of applied systems analysis* 16.1 (1989), pp. 3–9.

[16] Jennifer Rowley. "The wisdom hierarchy: representations of the DIKW hierarchy". In: *Journal of information science* 33.2 (2007), pp. 163–180.

[17] Gene Bellinger, Durval Castro, and Anthony Mills. *Data, information, knowledge, and wisdom*. 2004.

[18] Open Knowledge Foundation. *What is open?* URL: `https://okfn.org/opendata/` (visited on 01/21/2021).

[19] Yannis Charalabidis, Anneke Zuiderwijk, Charalampos Alexopoulos, Marijn Janssen, Thomas Lampoltshammer, Enrico Ferro, et al. "The World of Open Data". In: *Public Administration and Information Technology. Cham: Springer International Publishing. doi* 10 (2018), pp. 978–3.

[20] Open Data Inception. *2600+ Open Data Portals Around the World*. URL: `https://opendatainception.io` (visited on 01/21/2021).

[21] Wendy Carrara, Wae-San Chan, Sander Fischer, and E van Steenbergen. "Creating value through open data: Study on the impact of re-use of public data resources". In: *European Commission* (2015).

[22] European Publication Office. *EU Datathon*. URL: `https://op.europa.eu/en/web/eudatathon` (visited on 01/21/2021).

[23] Christian Bizer, Tom Heath, and Tim Berners-Lee. "Linked data: The story so far". In: *Semantic services, interoperability and web applications: emerging concepts*. IGI Global, 2011, pp. 205–227.

[24] Tim Berners-Lee. *Linked data-design issues*. URL: `http://www.w3.org/DesignIssues/LinkedData.html` (visited on 01/21/2021).

[25] Dave Beckett and Brian McBride. "RDF/XML syntax specification (revised)". In: *W3C recommendation* 10.2.3 (2004).

[26] Oxford Learners Dictionaries. *Metadata*. URL: `https://www.oxfordlearnersdictionaries.com/definition/english/metadata` (visited on 01/21/2021).

[27] Sebastian Neumaier, Lörinc Thurnay, Thomas J Lampoltshammer, and Tomá Knap. "Search, filter, fork, and link open data: The adequate platform: data-and community-driven quality improvements". In: *Companion Proceedings of the The Web Conference 2018*. 2018, pp. 1523–1526.

[28] Inc. Gartner. *Gartner Predicts that Organizations Using Blockchain Smart Contracts Will Increase Overall Data Quality by 50%*. 2020. URL: `https://www.gartner.com/en/newsroom/press-releases/2020-01-30-gartner-predicts-that-organizations-using-blockchain-` (visited on 01/21/2021).

[29] Igor Zikratov, Alexander Kuzmin, Vladislav Akimenko, Viktor Niculichev, and Lucas Yalansky. "Ensuring data integrity using blockchain technology". In: *2017 20th Conference of Open Innovations Association (FRUCT)*. IEEE. 2017, pp. 534–539.

[30] Xueping Liang, Sachin Shetty, Deepak Tosh, Charles Kamhoua, Kevin Kwiat, and Laurent Njilla. "Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability". In: *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. IEEE. 2017, pp. 468–477.

[31] Vincent Julien David, Yogaraj Jayaprakasam, Daniel Reznik, Hemant Bhatia, Travis Brown, and Ashwin Nagalla. *Data enrichment environment using blockchain*. US Patent 10,740,492. Aug. 2020.

[32] Jan Van Leeuwen and Jan Leeuwen. *Handbook of theoretical computer science: Algorithms and complexity*. Vol. 1. Elsevier, 1990.

[33] Christof Paar and Jan Pelzl. *Understanding cryptography: a textbook for students and practitioners*. Springer Science & Business Media, 2009.

[34] Bart Preneel. “The state of cryptographic hash functions”. In: *School organized by the European Educational Forum*. Springer. 1998, pp. 158–182.

[35] Andreas M Antonopoulos. *Mastering Bitcoin: Programming the open blockchain*. " O’Reilly Media, Inc.", 2017.

[36] Cameron F Kerry and Patrick D Gallagher. “Digital signature standard (DSS)”. In: *FIPS PUB* (2013), pp. 186–4.

[37] John E Canavan. *Fundamentals of network security*. Artech House, 2001.

[38] Werner Vogels. “Eventually consistent”. In: *Queue* 6.6 (2008), pp. 14–19.

[39] Seth Gilbert and Nancy Lynch. “Perspectives on the CAP Theorem”. In: *Computer* 45.2 (2012), pp. 30–36.

[40] Michel Rauchs, Apolline Blandin, Keith Bear, and Stephen B McKeon. “2nd Global Enterprise Blockchain Benchmarking Study”. In: *Available at SSRN 3461765* (2019).

[41] Iuon-Chang Lin and Tzu-Chun Liao. “A survey of blockchain security issues and challenges.” In: *IJ Network Security* 19.5 (2017), pp. 653–659.

[42] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Xiangping Chen, and Huaimin Wang. “Blockchain challenges and opportunities: A survey”. In: *International Journal of Web and Grid Services* 14.4 (2018), pp. 352–375.

[43] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. Tech. rep. Manubot, 2019.

[44] Ujan Mukhopadhyay, Anthony Skjellum, Oluwakemi Hambolu, Jon Oakley, Lu Yu, and Richard Brooks. “A brief survey of cryptocurrency systems”. In: *2016 14th annual conference on privacy, security and trust (PST)*. IEEE. 2016, pp. 745–752.

[45] Vitalik Buterin et al. *Ethereum white paper*. URL: `https://ethereum.org/en/whitepaper/` (visited on 01/21/2021).

[46] Christian Welzel, Klaus-Peter Eckert, Fabian Kirstein, and Volker Jacumeit. “Mythos Blockchain: Herausforderung für den öffentlichen Sektor”. In: *Fraunhofer FOKUS, Berlin* 21.1 (2017).

[47] Karl Wüst and Arthur Gervais. “Do you need a blockchain?” In: *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*. IEEE. 2018, pp. 45–54.

[48] Evangelos Deirmentzoglou, Georgios Papakyriakopoulos, and Constantinos Patsakis. “A survey on long-range attacks for proof of stake protocols”. In: *IEEE Access* 7 (2019), pp. 28712–28725.

[49] Shijie Zhang and Jong-Hyouk Lee. "Analysis of the main consensus protocols of blockchain". In: *ICT Express* 6.2 (2020), pp. 93–97.

[50] Nicolas T Courtois. "On the longest chain rule and programmed self-destruction of crypto currencies". In: *arXiv preprint arXiv:1405.0534* (2014).

[51] Wenbo Wang, Dinh Thai Hoang, Zehui Xiong, Dusit Niyato, Ping Wang, Peizhao Hu, and Yonggang Wen. "A survey on consensus mechanisms and mining management in blockchain networks". In: *arXiv preprint arXiv:1805.02707* (2018), pp. 1–33.

[52] Sarwar Sayeed and Hector Marco-Gisbert. "Assessing blockchain consensus and security mechanisms against the 51% attack". In: *Applied Sciences* 9.9 (2019), p. 1788.

[53] Daniel Fullmer and A Stephen Morse. "Analysis of difficulty control in bitcoin and proof-of-work blockchains". In: *2018 IEEE Conference on Decision and Control (CDC)*. IEEE. 2018, pp. 5988–5992.

[54] Arthur Gervais, Ghassan O Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. "On the security and performance of proof of work blockchains". In: *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 2016, pp. 3–16.

[55] Nicholas Stifter, Aljosha Judmayer, Philipp Schindler, Alexei Zamyatin, and Edgar Weippl. "Agreement with satoshi–on the formalization of nakamoto consensus". In: (2018).

[56] Vitalik Buterin and Virgil Griffith. "Casper the friendly finality gadget". In: *arXiv preprint arXiv:1710.09437* (2017).

[57] Iddo Bentov, Rafael Pass, and Elaine Shi. "Snow White: Provably Secure Proofs of Stake." In: *IACR Cryptol. ePrint Arch.* 2016 (2016), p. 919.

[58] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. "Ouroboros: A provably secure proof-of-stake blockchain protocol". In: *Annual International Cryptology Conference*. Springer. 2017, pp. 357–388.

[59] Daniel Larimer. "Delegated proof-of-stake (dpos)". In: *Bitshare whitepaper* (2014).

[60] Cong T Nguyen, Dinh Thai Hoang, Diep N Nguyen, Dusit Niyato, Huynh Tuong Nguyen, and Eryk Dutkiewicz. "Proof-of-stake consensus mechanisms for future blockchain networks: fundamentals, applications and opportunities". In: *IEEE Access* 7 (2019), pp. 85727–85745.

[61] Stefano De Angelis, Leonardo Aniello, Roberto Baldoni, Federico Lombardi, Andrea Margheri, and Vladimiro Sassone. "Pbft vs proof-of-authority: applying the cap theorem to permissioned blockchain". In: (2018).

[62] Parinya Ekparinya, Vincent Gramoli, and Guillaume Jourjon. "The attack of the clones against proof-of-authority". In: *arXiv preprint arXiv:1902.10244* (2019).

[63] Miguel Castro, Barbara Liskov, et al. "Practical Byzantine fault tolerance". In: *OSDI*. Vol. 99. 1999. 1999, pp. 173–186.

[64] Miguel Castro and Barbara Liskov. "Practical Byzantine fault tolerance and proactive recovery". In: *ACM Transactions on Computer Systems (TOCS)* 20.4 (2002), pp. 398–461.

[65] Roberto Saltini and David Hyland-Wood. "IBFT 2.0: A Safe and Live Variation of the IBFT Blockchain Consensus Protocol for Eventually Synchronous Networks". In: *arXiv preprint arXiv:1909.10194* (2019).

[66] Kostis Karantias, Aggelos Kiayias, and Dionysis Zindros. "Proof-of-burn". In: *International Conference on Financial Cryptography and Data Security*. Springer. 2020, pp. 523–540.

[67] David Cooper, Stefan Santesson, S Farrell, Sharon Boeyen, Rusell Housley, and W Polk. "RFC 5280: Internet X. 509 public key infrastructure certificate and certificate revocation list (CRL) profile". In: *IETF, May* (2008).

[68] Petra Wohlmacher. "Digital certificates: a survey of revocation methods". In: *Proceedings of the 2000 ACM workshops on Multimedia*. 2000, pp. 111–114.

[69] S Santesson, M Myers, R Ankney, A Malpani, S Galperin, and C Adams. *X. 509 internet public key infrastructure online certificate status protocol-OCSP (rfc 6960)*. Tech. rep. RFC 6960. RFC Editor. 1–41 pages, 2013.

[70] D Eastlake. "RFC 6066-Transport Layer Security (TLS) extensions: Extension definitions". In: *Technical report* (2011).

[71] E Rescorla. "Rfc 8446: The transport layer security (tls) protocol version 1.3". In: *Internet Engineering Task Force (IETF)* (2018).

[72] Antoine Delignat-Lavaud, Martín Abadi, Andrew Birrell, Ilya Mironov, Ted Wobber, and Yinglian Xie. "Web PKI: Closing the Gap between Guidelines and Practices." In: *NDSS*. 2014.

[73] CA/Browser Forum. *Baseline Requirementsfor the Issuance and Management of Publicly Trusted Certificates*. URL: https://cabforum.org/baseline-requirements-documents/ (visited on 10/08/2020).

[74] CA/Browser Forum. *Guidelines For The Issuance And Management Of Extended Validation Certificates*. URL: https://cabforum.org/extended-validation/ (visited on 10/08/2020).

[75] Deepak Kumar, Zhengping Wang, Matthew Hyder, Joseph Dickinson, Gabrielle Beck, David Adrian, Joshua Mason, Zakir Durumeric, J Alex Halderman, and Michael Bailey. "Tracking certificate misissuance in the wild". In: *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2018, pp. 785–798.

[76] J Ronald Prins and Business Unit Cybercrime. "DigiNotar Certificate Authority breach "Operation Black Tulip"". In: *Fox-IT, November* (2011), p. 18.

[77] Google Inc. *Certificate Transparency*. URL: https://www.certificate-transparency.org/ (visited on 10/08/2020).

[78] DigiCert. *Moving forward: What DigiCert's CT2 log retirement means for you*. URL: https://www.digicert.com/blog/digicert-statement-on-ct2-log/ (visited on 10/08/2020).

[79] Josh Aas and Sarah Gran. *Let's Encrypt Has Issued a Billion Certificates*. 2020. URL: https://letsencrypt.org/2020/02/27/one-billion-certs.html (visited on 01/18/2021).

[80] R Barnes, J Hoffman-Andrews, D McCarney, and J Kasten. "RFC 8555: Automatic Certificate Management Environment (ACME)". In: *Proposed Standard* (2019).

[81] J Callas, L Donnerhacke, H Finney, D Shaw, and R Thayer. "OpenPGP message format (RFC 4880)". In: *Internet Engineering Task Force* (2007).

[82] Michael Lucas. *PGP & GPG: Email for the practical paranoid*. No Starch Press, 2006.

[83] Cricket Liu and Paul Albitz. *DNS and Bind*. " O'Reilly Media, Inc.", 2006.

[84] Paul Mockapetris. *RFC 1035 - Domain Names - Implementation and Specification*. Tech. rep. 1987.

[85] Susan Thomson, Christian Huitema, Vladimir Ksinant, and Mohsen Souissi. "RFC 3596 - DNS Extensions to Support IP Version 6". In: *Internet Engineering Task Force (IETF)* (2003).

[86] P. Hallam-Baker, R. Stradling, and J. Hoffman-Andrews. "RFC 8659 - DNS Certification Authority Authorization (CAA) Resource Record". In: *Internet Engineering Task Force (IETF)* (2019).

[87] Roy Arends, Rob Austein, Matt Larson, Dan Massey, and Scott Rose. "RFC 4034 - Resource records for the DNS security extensions". In: *Internet Engineering Task Force (IETF)* (2005).

[88] P Hoffman and J Schlyter. "RFC 6698 - The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA". In: *Internet Engineering Task Force (IETF)* (2012).

[89] Suranjith Ariyapperuma and Chris J Mitchell. "Security vulnerabilities in DNS and DNSSEC". In: *The Second International Conference on Availability, Reliability and Security (ARES'07)*. IEEE. 2007, pp. 335–342.

[90] Eric Osterweil. "A Cybersecurity Terminarch: Use It Before We Lose It". In: *IEEE Security & Privacy* 18.4 (2020), pp. 67–70.

[91] Jonathan Weinberg. "ICANN and the Problem of Legitimacy". In: *Duke LJ* 50 (2000), p. 187.

[92] Jack Goldsmith. "Who controls the Internet? Illusions of a borderless world". In: *Strategic Direction* (2007).

[93] Frederic Cambus. *StatDNS - DNS and Domain Name statistics*. URL: `https://www.statdns.com/` (visited on 12/08/2020).

[94] Denic. *DNSSEC Domain Statistik*. URL: `https://www.denic.de/wissen/statistiken/monatsauswertung-dnssec/` (visited on 12/08/2020).

[95] Taejoong Chung, Roland van Rijswijk-Deij, David Choffnes, Dave Levin, Bruce M Maggs, Alan Mislove, and Christo Wilson. "Understanding the role of registrars in DNSSEC deployment". In: *Proceedings of the 2017 Internet Measurement Conference*. 2017, pp. 369–383.

[96] Spencer Roth, Roland van Rijswijk-Deij, and Taejoong Chung. "Tracking Registrar Support for DNSSEC". In: (2019).

[97] P Wouters. "RFC 7929 -DNS-Based Authentication of Named Entities (DANE) Bindings for OpenPGP". In: *Internet Engineering Task Force (IETF)* (2016).

[98] Eric Osterweil, Burt Kaliski, Matt Larson, and Danny McPherson. "Reducing the X. 509 Attack Surface with DNSSEC's DANE". In: *Securing and Trusting Internet Names, SATIN* 12 (2012).

[99] Ueli Maurer. “Modelling a public-key infrastructure”. In: *European Symposium on Research in Computer Security*. Springer. 1996, pp. 325–350.