universität wien

MASTERARBEIT | MASTER'S THESIS

Titel | Title

ArchiveChain: Towards a Blockchain based audit-proof electronic archive

verfasst von | submitted by

Felix Marcial B.Eng.

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien | Vienna, 2025

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 066 926
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Masterstudium Wirtschaftsinformatik
Betreut von | Supervisor:	ao. Univ.-Prof. i.R. tit. Univ.-Prof. Dipl.-Ing. Dr. Erich Schikuta

Abstract

Due to the growing number of digital business processes across diverse industries, the volume of electronically created and processed documents continues to grow. By law, businesses in the European Union must ensure that documents and records are archived in an audit-proof manner to comply with various retention requirements, particularly in areas such as taxation, human resources, accounting, and finance. In case of a tax audit, correctly archived accounting documents and records, such as receipts, invoices, and business documents, can provide the necessary evidence. Non-compliant archiving can lead to legal disadvantages and loss of evidence, resulting in costly consequences in legal disputes. The literature shows that the use of blockchain technology offers an excellent opportunity to ensure data integrity in systems that need to preserve information. Although blockchain is becoming increasingly popular in records management, compliance with regard to archiving has not been addressed in previous solutions. This thesis examines how blockchain technology can be used to support the requirements of an audit-proof archiving system according to the regulations applicable in Germany. A proof-of concept prototype is implemented using a private, permissioned blockchain that serves as an integrity layer for the archival system.

In the context of this work, integrity is ensured by storing files off-chain while the metadata and hash values of those files are recorded immutably on the blockchain. Access rights are transparently managed through smart contracts, ensuring that archived data is available only to authorized users. Additionally, the on chain metadata is used to automatically enforce legal retention periods. The system supports essential archiving functions such as the upload, authorized retrieval, and proper deletion of documents. The document-related interactions are immutably logged on the blockchain, supporting traceability and verifiability throughout the archiving process.

Overall, the results indicate that blockchain technology can effectively support several core requirements for an audit-proof archiving system and holds considerable promise for further enhancements.

Kurzfassung

Aufgrund der zunehmenden Anzahl digitaler Geschäftsprozesse in verschiedensten Branchen steigt auch die Menge an elektronisch erstellten und verarbeiteten Dokumenten stetig. Gesetzlich sind Unternehmen in der Europäischen Union dazu verpflichtet, Dokumente und Aufzeichnungen revisionssicher zu archivieren, um verschiedene Aufbewahrungspflichten zu erfüllen. Dazu gehören insbesondere Bereiche wie das Steuerwesen, Personalwesen, der Buchhaltung sowie das Finanzwesen. Im Falle einer Betriebsprüfung können korrekt archivierte Buchhaltungsunterlagen und Aufzeichnungen wie Belege, Rechnungen und Geschäftsunterlagen den notwendigen Nachweis erbringen. Nicht konforme Archivierung kann zu rechtlichen Nachteilen und dem Verlust von Beweismitteln führen, was in Rechtsstreitigkeiten hohe Kosten nach sich ziehen kann. Aus der Literatur geht hervor, dass der Einsatz von Blockchain-Technologie eine hervorragende Möglichkeit bietet, die Datenintegrität in Systemen mit langfristigem Aufbewahrungsbedarf sicherzustellen. Obwohl die Blockchain im Bereich des Dokumentenmanagements immer beliebter wird, wurden die Anforderungen an die Compliance, speziell für die Archivierung, in bisherigen Lösungen nicht berücksichtigt. In dieser Arbeit wird untersucht, wie die Blockchain-Technologie eingesetzt werden kann, um die Anforderungen an ein revisionssicheres Archivierungssystem gemäß den in Deutschland geltenden Vorschriften zu unterstützen. Zu diesem Zweck wurde ein Proof of Concept entwickelt, das eine private, zugriffsbeschränkte Blockchain als Integritätsschicht für das Archivsystem verwendet.

Im Rahmen dieser Arbeit wird die Integrität durch die Speicherung von Dateien außerhalb der Blockchain gewährleistet, während die Metadaten und Hash-Werte dieser Dateien unveränderlich auf der Blockchain gespeichert werden. Zugriffsrechte werden transparent über Smart Contracts verwaltet, sodass nur autorisierte Nutzer auf Archivdaten zugreifen können. Darüber hinaus dienen die On-Chain-Metadaten der automatisierten Durchsetzung gesetzlicher Aufbewahrungsfristen. Das System unterstützt dabei zentrale Archivierungsfunktionen wie den Upload, den autorisierten Abruf und die ordnungsgemäße Löschung archivierter Dateien. Die dokumentrelevanten Interaktionen werden dabei manipulationssicher auf der Blockchain protokolliert und tragen so zur Nachvollziehbarkeit und Prüfbarkeit bei.

Insgesamt zeigen die Ergebnisse, dass die Blockchain-Technologie verschiedene zentrale Anforderungen an ein revisionssicheres elektronisches Archiv effektiv erfüllen kann und dabei vielversprechendes Potenzial für zukünftige Erweiterungen bietet.

Contents

List of Tables

List of Figures

Listings

1. Introduction

With the ongoing digital transformation, the global volume of data is increasing exponentially. In the business context, the growing digitalization, along with the rising number of organizational policies, governmental laws, and regulations regarding data retention, is leading to an increasing demand for long-term storage solutions [RMK18].In Germany, electronic documents such as tax records, receipts, and business reports are subject to legal retention requirements that companies must comply with. The key term here is audit-proof archiving, which is mandatory for companies in Germany to ensure compliance. Adhering to these regulations is essential, as audit-proof archived documents can serve as proof of the legality of business operations, for example, in the event of a tax or operational audit. Failure to meet archiving obligations can result in significant legal and financial consequences for companies. This presents both technical and organizational challenges. Companies must ensure that documents remain unchanged for years, that all access and potential modifications are transparently recorded, and that archived content is kept in readable and accessible form during the duration of the retention period. In practice, companies often use centralized storage systems for audit-proof archiving, following the "Write Once, Read Many" (WORM) principle to protect stored data from subsequent modifications [THLG18]. In addition to on-premise solutions, cloud-based storage is gaining increasing importance, as it offers advantages in terms of availability, scalability, and cost efficiency. Although these systems ensure a high level of fault tolerance, risks still remain. In a worst-case scenario, archived documents could be accidentally modified, maliciously manipulated, or even deleted unnoticed [RMK18].

One technology that is increasingly being discussed in the context of archiving is blockchain [PMTM22]. As the most well-known form of Distributed Ledger Technology (DLT), blockchain enables immutable and transparent data storage. Unlike centralized solutions, where a single entity is responsible for managing and securing data, blockchain operates on a decentralized network. Data is stored in blocks that are cryptographically linked to each other. Each transaction is recorded in a new block, which is connected to the previous block via a hash value. This structure ensures that any subsequent modification would affect the entire chain, making manipulations immediately detectable. Due to this mechanism, blockchain is particularly suitable for use cases where data integrity and traceability are essential [WG18a]. A key feature of many blockchain platforms is their programmability through Smart Contracts, which are self-executing programs that automatically run when predefined conditions are met [RMK18]. They offer a programming interface for interacting with the underlying blockchain storage models, enabling logic for tasks like state management, governance enforcement, and credential verification [BLS+20]. This offers potential for their use in archive-related functions.

Despite the recognizable advantage for archiving, there has been little research so far

on meeting regulatory requirements using blockchain technology. This study therefore analyzes the extent to which blockchain technology can support the legal requirements for audit-proof archiving in Germany. According to the research questions, the expected deliverables of the thesis are as follows. First, an analysis of regulatory requirements for audit-proof archiving, focusing on relevant legal frameworks and standards applicable in Germany. Second, a concept for integrating blockchain technology into an archival system. The concept examines different architectural models and explores how blockchain can be utilized in the context of audit-proof archiving. Third, a proof of concept (PoC) demonstrating the feasibility of the proposed approach. The PoC serves to validate the conceptual model in a practical setting and implements key functionalities such as the upload, retrieval, and deletion of archival data, leveraging blockchain and smart contracts to ensure compliance with the legal requirements. Finally, an evaluation of the blockchain-based archiving approach against the defined requirements. This includes an assessment of technical feasibility, potential limitations, and challenges, as well as an outline of areas for future research.

Thesis Structure

Chapter 2 provides the theoretical foundation and discusses the state of the art. It defines electronic archiving and outlines regulations and standards for audit-proof archiving in Germany. Additionally, it introduces blockchain as a distributed ledger technology, covering its key aspects and applications in record-keeping. Building on these insights, Chapter 3 briefly summarizes the findings on regulatory requirements and blockchain-based record-keeping, leading to the formulation of the research questions. In chapter 4, the key findings from the state of the art are derived into concrete requirements that the system needs to fulfill. Chapter 5 then delves into the conceptual design of the system, outlining the architectural framework and presenting a proof of concept that demonstrates how blockchain technology can contribute to audit proof archiving requirements. This is followed by Chapter 6, which discusses the system's technology stack, covering the selection of blockchain and off-chain storage solutions, along with an overview of key dependencies used in the implementation. With the technical foundation established, Chapter 7 details the implementation process, focusing on system setup and the realization of the blockchain-based archival solution. To demonstrate the system's practical application, Chapter 8 introduces three use cases that highlight its functionality and benefits in a business scenario. Chapter 9 evaluates the developed solution by critically assessing how well the implemented system meets the defined requirements. Additionally, it discusses the limitations of the prototype and suggests potential improvements and future research directions. Finally, Chapter 10 concludes the thesis.

2. Foundations and State of the Art

This chapter provides an overview of the terms and topics relevant to this thesis. It explains different types of electronic archiving, with a focus on audit-proof archiving and its regulatory framework. It also introduces blockchain technology as a distributed ledger, including its fundamentals and its potential for record-keeping.

2.1. Definitions

Electronic archiving is a key aspect of records management, which governs the creation, maintenance, storage, and disposition of records, including electronic information. In Germany, a fundamental distinction is made between three types of electronic archiving [UK97].

2.1.1. Electronic archiving

Kampffmeyer defines the term Electronic Archiving as follows:

> "The term electronic archiving stands for the unchangeable, long-term retention of electronic information."

For this purpose, electronic archiving systems are typically used. However, the scientific definition of an archive and archiving differs in content from the one used in the document management industry [Kam12].

2.1.2. Electronic long-term archiving

The term 'archiving' inherently implies longevity. However, the concept of 'long-term archiving' specifically refers to electronic archiving systems which ensure that documents are retained for at least seven years or more in accordance with the legal retention periods defined by the German Commercial Code (HGB) and German Fiscal Code (AO) [UK97].

2.1.3. Audit-proof archiving

Audit-proof essentially means that the systems in place are designed in a way that ensures their compliance with legal requirements can be verified. It is a specific type of electronic archiving in which business and tax-related documents in Germany must be stored in compliance with legal requirements. The term "audit- proof archiving" is not clearly defined by law. It was first introduced by Ulrich Kampffmeyer in 1992 and later published

in a Code of Practice by the VOI (Association for Optical Information Systems) in 1996. Due to the growing need for legally compliant archives, the concept of audit-proof archiving has become widely adopted in the industry. The word "audit" originates from Latin and is commonly understood as "review" or "inspection." In this context, audit-proof archiving refers to the ability to retrospectively verify compliance with legal requirements. This concept extends beyond just the technical solution. It also involves the formal approval of the entire process, organizational structure, secure workflows, and documented proof of compliance through procedure documentation [Kam12]. The VOI Code of Practice [Org09] defines voluntary rules of conduct from the general legal requirements in the German Commercial Code (HGB) and German Fiscal Code (AO), for creating an audit-proof electronic archiving system. These were summarized in ten specific guidelines [Kam02].

- Every document must be preserved in accordance with legal and organizational requirements.
- Archiving must be complete – no document may be lost on its way to the archive.
- Each document must be archived at the earliest possible organizational point.
- Every document must match its original and be archived in an unalterable form.
- Each document may only be accessed by appropriately authorized users.
- Every document must be searched and displayed within a reasonable time.
- No document may be destroyed, i.e., deleted from the archive, before the end of its retention period.
- Any modifying action in the electronic archive system must be logged in a way that is traceable for authorized individuals.
- The entire organizational and technical archiving process can be audited by a qualified third party at any time.
- All migrations and changes to the archive system must ensure compliance with all previously listed principles.

Kampffmeyer describes the guidelines and recommendations of the VOI as a way to achieve legal certainty [KM97], but provides a more detailed definition:

> "Audit-proof archiving refers to an archiving system solution that complies with the requirements of the HGB §§239, 257, as well as the AO and the GoBS, ensuring the secure and proper retention of commercial documents while meeting the required retention periods of six to ten years." [Kam03]

The GoBS represents the principles of proper IT-supported accounting systems, which were summarized and replaced by the tax authorities in 2014 with the publication of the Principles for the proper keeping and retention of books, records, and documents in electronic form and for data access (GoBD). The GoBD define how IT-supported accounting is to be properly designed from the perspective of the tax authorities [dF]. It also includes audit-proof archiving [BBSBBS21], and can therefore serve as a guideline for companies to meet compliance requirements [HKG16].

Companies that fail to comply with regulatory requirements may be subject to substantial fines. Audit-proof archiving is thus an essential component of compliance for information systems. While the term was originally used for the areas of commercial and tax law, it is now considered synonymous with the general archiving of information [Con09].

2.2. Regulations & Standards

The German Commercial Code (§ 239, § 257) and the Fiscal Code (§ 146, § 147, § 200) provide the legal basis for storing documents in electronic systems. These regulations are interpreted in the tax law context through the GoBD. When the Federal Ministry of Finance introduced the GoBD in 2014, the goal was to establish uniform regulations and bring more clarity to tax compliance requirements [HKG16]. The GoBD define the principles of proper accounting for electronic bookkeeping and require compliance with specific regulatory criteria. These criteria outline the general requirements for tax relevant IT systems, covering all electronic applications used to create, process, archive, or transmit data. In essence, the GoBD sets principles for the electronic storage of information and the proper handling of tax-relevant documents. However, due to short innovation cycles and a technology-neutral approach, the GoBD does not provide specific implementation guidelines [BI]. As a result, companies often face uncertainty when interpreting the regulations. It is therefore recommended to refer to the publications of the Institute of Public Auditors in Germany, Incorporated Association (IDW), as guidance for the practical implementation of the GoBD [BL24].

The IDW RS statements on accounting issued by the Expert Committee for Information Technology (FAIT) are specifically referenced in practical guidelines [fwVe, GB17], industry publications [BI], and technical contributions from tax associations and advisory chambers [Bun] as guidance for implementing the requirements. When using an electronic archiving system, particular attention should be given to IDW RS FAIT 3 – Principles of Proper Accounting When Using Electronic Archiving Procedures [GB17]. The following list provides an overview of the IDW Statements on Accounting:

- IDW RS FAIT 1 – Principles of Proper Accounting When Using Information Technology (2002)
- IDW RS FAIT 2 – Principles of Proper Accounting When Using Electronic Commerce (2003)

- IDW RS FAIT 3 – Principles of Proper Accounting When Using Electronic Archiving Procedures (2015)
- IDW RS FAIT 4 – Requirements for the Properness and Security of IT-Supported Consolidation Processes (2012)
- IDW RS FAIT 5 – Principles of Proper Accounting When Outsourcing Accounting-Relevant Processes and Functions, Including Cloud Computing (2015)

Both the GoBD and IDW RS FAIT define compliance requirements, which can be collectively referred to as "IT compliance" [DD21]. According to [KD08], this term describes a state in which a company demonstrably meets all relevant legal requirements in the IT sector. This thesis focuses not only on the GoBD but also on the specifications of the IDW RS FAIT 3, as the archiving system is the central aspect in this context [Däs14]. These are described in more detail in the following section to provide an overview of the necessary requirements for the system to be designed.

2.2.1. GoBD

With the letter of the Federal Ministry of Finance on 14. November 2014, the principles for the proper keeping and retention of books, records and documents in electronic form as well as for data access (GoBD) from the German Commercial Code and the German Fiscal Code were summarized. These guidelines, which originally replaced the former principles of proper IT-based accounting systems (GoBS) and principles of data access and verifiability of digital documents (GDPdU), were further specified and extended with the new version of November 28, 2019, while retaining the structural basis of the original version. The GoBD are not only important for large companies, but affect almost all company sizes and forms, as hardly any company today practices exclusively paper-based accounting [Kli20]. In addition to the management of data, archiving is also described, which in Germany must be carried out according to specific characteristics. The regulations aim to define how companies create, store, access, preserve and archive information in different forms for increasingly long periods of time [UK97]. The minimum requirements of the GoBD are the general compliance requirements, which are derived from both commercial law regulations (HGB) and tax regulations (AO) [dF].

- **Principle of traceability and verifiability** (§ 145 Subsection 1 AO, § 238 Subsection 1 sentence 2 and sentence 3 HGB),
- **Principles of truth, clarity and continuous recording:**
  - **Completeness** (§ 146 Subsection 1 AO, § 239 Subsection 2 HGB),
  - **Correctness** (§ 146 Subsection 1 AO, § 239 Subsection 2 HGB),
  - **Timely bookings and recording** (§ 146 Subsection 1 AO, § 239 Subsection 2 HGB),
  - **Order** (§ 146 Subsection 1 AO , § 239 Subsection 2 HGB),

– **Immutability** (§ 146 Subsection 4 AO, § 239 Subsection 3 HGB)

These principles must be demonstrably fulfilled and maintained for the duration of the retention period [dF]. Based on the GoBD, the following additional requirements must be observed in addition to the general compliance requirements [RG15]:

- **Internal Control System**
- **Data Security**
- **Retention** (§ 147 Subsection 1 AO),
- **Machine readability** (§ 147 Subsection 2 AO),
- **Data Access** (§ 147 Subsection 6 AO),

The GoBD does not specify any direct implementation guidelines for the requirements, which is justified by the fact that technical specifications or standards, especially regarding archival media or cryptographic methods, cannot be established due to rapid technological developments. [dF].

Principle of traceability and verifiability

The principle of traceability and verifiability requires that all business transactions and the associated accounting and recording procedures are designed and documented in such a way that they are understandable and verifiable for a knowledgeable third party, both in terms of their origin and their processing [fwVe, dF]. This includes the complete tracking of business transactions, from the original document to the final posting in the annual financial statements and vice versa. This progressive and retrospective audit capability ensures the integrity of the entire accounting system and ensures that company management and external auditors, such as tax consultants or auditors, can obtain a clear overview of business activities.

If, for example, a receipt cannot be assigned to a corresponding booking, the bookkeeping may be considered non-compliant due to a lack of completeness [Lee20], meaning that the requirements for traceability are not met. When using an electronic archiving system, it must be ensured that both the storage of data and the processes involved remain traceable throughout the entire retention period. The Federal Ministry of Finance (BMF) also emphasizes the importance of creating a comprehensive and meaningful procedure documentation, which describes the entire lifecycle of a business transaction, from its creation to its retention. In addition to documenting the processes within the archiving system, this also includes a description of how the general compliance requirements are met [fwVe].

Principles of truth, clarity and continuous recording

Completeness

The principle of completeness requires that all business transactions and the associated documents and records are recorded in full and without gaps [dF]. This is to ensure that all relevant information on financial transactions and operational processes is available for proper accounting and its verification. For recording, the individual recording obligation applies, meaning that each business transaction must be recorded separately and may not be taken into account more than once [fwVe].

Correctness

According to the principle of correctness (§ 146 Subsection 1 AO), receipts, books, and records must accurately reflect business transactions as they actually occurred and in compliance with legal regulations. They must be recorded truthfully and assigned properly to the appropriate accounts [dF, fwVe]. Accordingly, business transactions may not simply be changed or created if they never took place.

Timely bookings and recording

The principle of timely booking and recording states that every business transaction must be properly documented and recorded as soon as it occurs. This rule ensures that transactions are recorded promptly and correctly in order to maintain the correctness of financial reporting [dF]. Additionally, this criterion helps prevent potential manipulations before archiving [Lee20]. It may include both organizational regulations and technical measures to ensure that data is promptly transferred to the electronic data processing system.

Order

The principle of order requires that documents and records be systematically recorded and stored in a clear, unambiguous, and traceable manner. Documents cannot simply be saved locally in a folder but must be stored according to specific organizational principles, which, however, are not explicitly defined in the GoBD [dF]. In practice, electronic filing structures and unique document identification should be used to ensure that business transactions can be easily located and traced [Lee20].

Immutability

The principle of immutability states that companies may not subsequently change their records in such a way that the original content can no longer be determined [dF]. However, changes can also be prohibited in the system [BI]. When using an electronic archiving system, appropriate measures should be implemented at the hardware, software, or organizational level to ensure data immutability. The letter from the Federal Ministry of

Finance on the GoBD specifies that logging changes and deletions [dF] are a prerequisite to ensure that data cannot be suppressed, overwritten, deleted, or falsified unnoticed [BI].

Internal Control System

The principles of proper accounting under § 146 AO require companies to establish an internal control system (ICS) to ensure the integrity and correctness of data. Such a system must include various control mechanisms, ranging from access and authorization controls to segregation of duties and data entry validation checks, all of which must be properly logged. These measures are designed to ensure that only authorized personnel have access to relevant information and that all data is recorded correctly and completely [dF]. The design and implementation of an ICS strongly depend on the organizational structure, complexity, technology, and business processes of the respective company [dF]. An effective ICS is also an integral part of procedure documentation, which outlines how the controls within the system are implemented and executed [BI].

Data Security

Data security is also a requirement of the GoBD. The archiving system must be protected against the risk of data loss, including safeguards against unauthorized entries and modifications through access and authorization controls. Insufficiently protected data is considered a violation of the legal regulations, which makes the bookkeeping formally non-compliant and potentially leads to legal consequences [dF]

Retention

According to the principles of proper accounting, all electronic data and documents subject to retention requirements must be stored completely and unaltered for the legally prescribed retention periods. This applies to both internally created documents and those received from third parties, whereby the data must be retained in the format in which they were originally created or received by the company. The documents listed in § 147 Subsection 1 AO represent the core records relevant for taxation under the GoBD. However, there is no universally applicable list, as the diversity of IT systems and company-specific software and hardware environments influence what is considered retention relevant [fwVe]. § 147 Subsection 1 AO specifies the following documents subject to retention:

- Books and records
- Inventories
- Annual financial statements
- Management reports
- Opening and closing balance sheets

- Accounting documents
- Commercial & business letters

As a general rule, retained documents must be stored within Germany, as specified in § 146 Subsection 2 Sentence 1 AO. However, storage abroad may be permitted with approval from the tax authorities. Retention periods vary depending on the document type, for example electronic invoices in Germany must be kept for eight years. Throughout the entire retention period, any modification or deletion of the stored data is strictly prohibited. According to § 147 Subsection 2 AO, documents subject to retention may be archived digitally, as long as their reproduction is identical to the original. This regulation allows documents to be stored electronically, provided they remain accessible at all times, can be made readable without delay, and are machine-processable." [fwVe]

Machine readability

According to § 147 Subsection 2 No. 2 AO, the tax authorities have the right to conduct a machine-based evaluation of data. Machine readability refers to the technical ability to systematically search and analyze all retained information throughout the entire legal retention period [BI]. This includes the capability to perform full-text searches, allowing data to be searched without predefined filters. The concept goes beyond simple data queries and also encompasses more complex operations, such as mathematical and technical analyses, screen-based queries, and the ability to track both internal and external links and references [fwVe].

Data Access

The GoBD specify the tax authorities' right to access all necessary company data during external tax audits. This access can be provided in different ways [dF]:

- Direct data access (Z1 access): The auditor is granted direct access to the company's IT systems, allowing them to independently and interactively retrieve and analyze stored data.
- Indirect data access (Z2 access): In this case, the taxable entity conducts specific evaluations as instructed by the tax authorities and provides the results to the auditors. This method enables the auditor to request specific information without directly accessing the systems.
- Provision of data storage media (Z3 access): The required data is transferred to the tax authorities on a physical or digital storage medium, allowing them to analyze the data independently.

If the required data access types are not provided, the external auditor may impose a payment penalty in accordance with § 146 Subsection 2 AO [fwVe].

2.2.2. IDW RS FAIT 3

The IDW RS FAIT 3 is a statement on accounting issued by the Institute of Public Auditors in Germany, Incorporated Association (IDW). The statement is titled "Principles of Proper Accounting When Using Electronic Archiving Procedures" and specifically defines the commercial law requirements for the electronic archiving of documents subject to retention [dWFfI06]. While these requirements help clarify legal obligations, they are not legally binding, as they merely interpret existing statutory duties [Lee20]. However, following these principles can reduce the risk of errors when complying with legal regulations. As a result, the guidelines can serve as a practical reference for companies to ensure compliance in electronic archiving.

The statement places particular emphasis on the design of the archiving system, which includes all components necessary for the capture, indexing, storage and management, retrieval, and long-term retention of electronic data and documents. In this context, the IDW identifies a combination of three key elements:

- the IT infrastructure used for archiving,
- the IT applications used for archiving, and
- the IT-supported archiving processes [dWFfI06].

According to IDW RS FAIT 3, the IT infrastructure typically includes an archive server, the system software installed on it, and a storage system for managing the storage media in use. Additionally, scanner units for digitizing image-based documents may also be part of the infrastructure. The IT applications generally consist of programs for data capture, such as scanning software, as well as for indexing and managing archived records and retrieving or displaying data. Finally, the IT-supported archiving processes describe the entire document and data flow, from the creation or receipt of relevant records to their immutable and long-term retention, and ultimately, their final disposal.

The IDW RS FAIT 3 shares significant overlaps with the GoBD. In addition to ensuring compliance with the general compliance requirements outlined in the GoBD, which are not reiterated here, it also specifies security requirements that must be observed for data and electronic documents [DD21]. Specifically, it defines detailed security criteria for electronic archiving procedures to ensure that the implemented systems comply with the principles of proper accounting (GoB) [dWFfI06]. The security criteria outlined in IDW RS FAIT 3 consist of the following elements:

- **Confidentiality**
- **Integrity**
- **Availability**
- **Authorization**
- **Authenticity**

- **Binding Character**

To ensure confidentiality, sensitive data must be protected from unauthorized access, disclosure, or distribution. The integrity requirement specifies that all data must be stored correctly, uniquely indexed, and protected against manipulation. Furthermore, continuous availability must be ensured throughout the entire retention period. Archiving systems, including the associated IT applications and infrastructure, should be designed to ensure that stored data and documents remain available for retrieval at all times. In practice, retrieval usually takes place through the online display of documents and data, the generation of printouts, or via an export interface. Furthermore, continuous availability must be guaranteed throughout the entire retention period. Archiving systems should be designed in such a way that data and documents remain retrievable for the entire retention period. Retrieval options include displaying data online, printing documents, or exporting them in a machine-readable form via an interface. The authorization requirement defines the measure ensuring that only authorized individuals can perform specific functions within the system and modify archived data and documents. Authenticity is maintained by establishing a clear connection between the original document and its digital copy, ensuring that each archived file can be linked to a specific business transaction. Finally, the requirement of binding character refers to the property of electronic archiving procedures to ensure the intended legal consequences. A deep understanding of these security objectives is essential for designing an audit proof system that can withstand long-term legal scrutiny.

2.3. Blockchain a Distributed Ledger Technology

DLT is a general protocol and structure that facilitates the decentralized management of data across a network of independent nodes. DLT eliminates the need for a central authority by employing advanced cryptographic techniques to link data securely [SZJS22]. This structure inherently supports environments where trust is minimal and interactions may occur among potentially unreliable or adversarial parties [RGG+18]. There are many variants of Distributed Ledger Technologies, with Blockchain being its most well-known representative [KSKS21]. It records and stores all transactions that take place chronologically across a peer-to-peer network. It is a continuously growing list of records, known as blocks, which are connected and protected using cryptographic methods. Each block usually includes a cryptographic hash referencing the previous block, along with a timestamp and a record of transactions [YCC19]. This establishes a linkage between the blocks, creating a chain, which is illustrated in Figure 2.1.

By design, a Blockchain is resistant to modification of data, which makes it especially useful in systems that need to preserve information. A Blockchain can be seen as a distributed database where every node has the same order of tamper-proof data. It uses a consensus algorithm in order to achieve consistency among the different nodes. [WEKJ17].

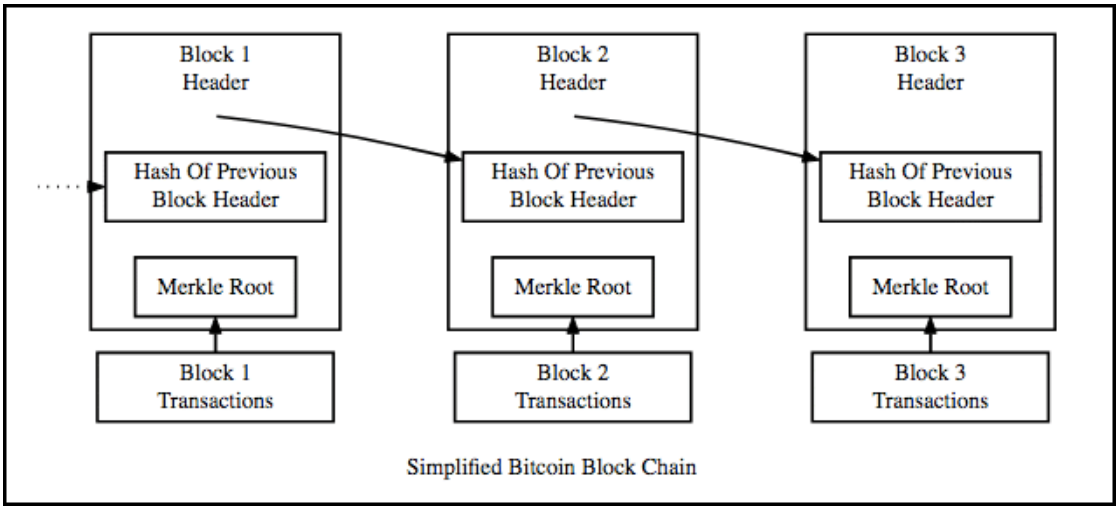

Figure 2.1.: Simplified Bitcoin Block Chain
[DS20]

2.3.1. Blockchain Characteristics

Blockchains have distinct characteristics that set them apart from traditional database technologies. Depending on the blockchain platform, certain aspects may be more or less pronounced. The most important characteristics include [YSJAH22, XYH18, JHW18, LL17, CCK+18, SC17, Mey22]:

Decentralization Blockchain operates on a decentralized network, which means it doesn't rely on a central point of control. Instead, it spreads out the data and decision-making across many computers or nodes. This structure prevents any single entity from having too much power or influence, enhancing security and reducing the risks of fraud or corruption. Decentralization also promotes resilience, as the system can continue operating even if parts of it fail.

Immutability Once data is entered into a blockchain, it cannot be altered or deleted. This feature is secured through cryptographic hashes, unique digital signatures that change completely if any data is tampered with. This characteristic ensures that all transactions recorded on a blockchain are permanent and tamper-proof, providing a trustworthy and unalterable history of data.

Data Provenance Blockchain provides a clear trail of data provenance, allowing users to trace where each piece of data originated and how it has been handled over time. Data provenance also increases accountability and transparency, as every transaction and its

origin are visible and verifiable by all participants.

Transparency While in a public blockchain, individual user identities can be protected through pseudonyms, the blockchain itself is highly transparent. In a public network, all participants on the network can view transactions and data entries in real-time. This level of openness helps build trust among users and can quickly expose any attempts at fraud, making it easier to maintain a fair and secure system.

Security In blockchain technology, security is enhanced through cryptography, decentralization, and consensus mechanisms. Every record on a blockchain is individually encrypted, while the decentralized structure minimizes the risk of data manipulation by removing any central point of failure. Blocks are cryptographically linked, with each block containing a unique hash and the hash of the previous block, forming a chain that is difficult to alter undetected. Consensus mechanisms require collaborative approval of all changes to the ledger, preventing fraudulent transactions and ensuring data consistency across network participants. Together, these measures protect the blockchain from unauthorized access and ensure its reliability and integrity.

Distributed Ledger A distributed ledger is a database that is jointly maintained and synchronized across multiple locations, institutions, or regions, and can be accessed by multiple participants. In blockchain, this ledger records all transactions across every node in the network, ensuring all copies are identical. This distributed nature helps eliminate single points of failure and enhances the security and accuracy of data stored on the blockchain [Mey22, CCK+18].

Programmability One of the core features of many blockchain platforms is their ability to support programmability through smart contracts. These are self-executing contracts in which the agreement terms are directly encoded. This functionality allows for the automated execution of agreements and transactions based on predefined rules and conditions, without the need for intermediaries. This characteristic is essential for the development and operation of decentralized applications (dApps) that are built on blockchain technology.

Consensus Consensus mechanisms are fundamental to maintaining the integrity and security of blockchain, as they enable distributed systems to agree on a single, consistent state of data. In blockchain, consensus models like Proof of Work (PoW) or Proof of Stake (PoS) ensure that all transactions are verified and agreed upon by all nodes before being permanently added to the ledger. This process prevents fraud and ensures that

each participant's ledger is kept in sync without the need for a central authority. In the next chapter, the most well-known consensus algorithms will be described in greater detail.

2.3.2. Consensus algorithms

According to Xiong (2022) [XCW+22], the consensus mechanism in blockchain technology ensures agreement among all network nodes regarding the current state of the network and the validity of transactions. This is essential for maintaining the security and integrity of the blockchain. Different blockchain platforms implement various consensus algorithms, such as Proof of Work (PoW), Proof of Stake (PoS), or Proof of Authority (PoA), to achieve consensus. However, the choice of the most suitable algorithm heavily depends on the specific use case of the blockchain. The Proof of Work (PoW) algorithm, used in the Bitcoin network [Nak08], requires miners to solve a complex computational problem to validate transactions and add them to the blockchain. This algorithm is intentionally energy-intensive and slow to make it costly and difficult for attackers to manipulate the chain. However, in practice, PoW often leads to a concentration of mining resources in a few large mining pools, increasing the risk of centralization and potential manipulation. Another major issue with PoW consensus is its high energy consumption, which is required to maintain the network and reach consensus. Additionally, scalability limitations exist, as PoW-based blockchains like Bitcoin, have a limited transaction capacity, making them slow and expensive under high network demand. This restricts their usability in scenarios requiring high-speed or large-scale transaction processing.

To address the high energy consumption of Proof of Work (PoW), Proof of Stake (PoS) provides a more energy-efficient alternative. Unlike PoW, where miners must use significant computing power to add new blocks to the blockchain, PoS operates on the principle of stake ownership. In this system, validators are chosen to verify transactions and add blocks based on the amount of assets they stake as collateral. This stake serves both as an incentive to maintain network integrity and as security against fraudulent behavior, as validators risk losing their deposit in case of manipulation attempts [ethc]. This collateral mechanism promotes honest behavior and strengthens trust in the network. Additionally, PoS enhances decentralization and reduces the risk of a 51% attack [AM+17], where a single entity could gain control over more than half of the network. Ethereum, the second-largest public blockchain by market capitalization, adopted PoS in 2023 and now requires a minimum of 16,384 validators, significantly increasing decentralization and security. With the transition to PoS, Ethereum also aims to address scalability issues. Before the upgrade, the network was capable of processing about 15 transactions per second. By introducing "Sharding", the network is divided into smaller parts that process and validate transactions in parallel, theoretically enabling up to 100,000 transactions per second. However, full Sharding capabilities are still several years away [etha].

Another popular alternative is Proof of Authority (PoA). Compared to other consensus algorithms, PoA offers better performance, butt at the expense of decentralization. It relies on the identity and reputation of individual validators to ensure network security. Unlike PoW and PoS, where anonymity often plays a role, validators in a PoA system are

publicly known and are selected based on their trustworthiness and reliability. Instead of staking assets, PoA validators put their reputation at risk, assuming that no more than one-third of them would act fraudulently. Due to its reliance on a small number of validators, PoA is better suited for private networks where participants are known, rather than public blockchains [Jos21].

The choice of a suitable consensus mechanism depends largely on the specific requirements of a blockchain network, as each mechanism has its own advantages and limitations. While Proof of Work (PoW) has traditionally been considered the established standard, Proof of Stake (PoS) and Proof of Authority (PoA) are gaining popularity due to their significant advantages in energy efficiency and scalability.

2.3.3. Types of Blockchain Systems

Blockchains can be categorized into different types, each serving different use cases and offering distinct advantages. An overview of the most common types and classifications is provided by [LL17, SY19].

- **Public Blockchain**: A public blockchain is fully decentralized and accessible to everyone. Anyone can conduct transactions, view data, and participate as a node to help secure the network. Since there is no central authority regulating access or controlling network operations, public blockchains are permissionless. This means that anyone can perform transactions or act as a validator without prior approval. Well-known examples include Bitcoin and Ethereum, which serve as the foundation for cryptocurrencies and decentralized applications (dApps).
- **Private Blockchain**: In contrast to public blockchains, private blockchains are closed networks, typically controlled by a single organization or institution. Since the identity and permissions of nodes are centrally managed, private blockchains are referred to as permissioned. Only authorized participants can participate in the consensus mechanism and validate transactions, ensuring greater control over the network.
- **Consortium/Federated Blockchain**: A consortium blockchain serves as a middle ground between public and private blockchains. Instead of being controlled by a single organization, it is managed by a group of companies or institutions, making it semi-decentralized. Only selected participants can validate transactions, increasing security while also allowing for greater scalability compared to public blockchains.
- **Hybrid Blockchain**: Hybrid blockchains combine characteristics of public, private, and consortium/federated blockchains. Organizations can determine which information is publicly visible and which remains private. This approach provides a flexible balance between transparency and privacy, making it suitable for use cases where a restricted participant base is required, yet public verification is still necessary. A specific example is e-voting, where only a predefined group of voters is allowed to participate, but the results must remain transparent and publicly verifiable [WG18b].

2.3.4. Data Storage in Blockchains

In blockchain technology, data storage is generally categorized into two methods: direct on-chain storage, where data is stored within the blockchain itself, and off-chain storage, where only a reference to an external storage location is kept on the blockchain.

On-Chain On-chain storage refers to storing data directly within the blockchain. Each transaction containing data becomes a permanent part of the blockchain, making it visible and verifiable by all participants. This method ensures high security and transparency, as data is validated and stored through blockchain's distributed ledger technology. However, on-chain storage has significant drawbacks, particularly regarding scalability and cost efficiency. Since the size of the blockchain increases linearly with each additional dataset, storing large amounts of data directly on-chain can quickly become impractical and costly [HSE+18].

Off-chain storage refers to a method where data is stored externally, and only a reference value, such as a cryptographic hash, is recorded on the blockchain. The hash acts as proof of immutability and validation, ensuring that the original data remains unaltered while being stored outside the blockchain. This method is beneficial for handling large or sensitive data and for lower storage costs. Off-chain storage solutions include traditional databases, distributed file systems such as the InterPlanetary File System [1] (IPFS), cloud storage services, and Distributed Hash Tables (DHTs) used in peer-to-peer networks. By leveraging off-chain storage, large datasets can be managed efficiently without the high costs and scalability limitations associated with on-chain storage [HSE+18].

2.3.5. Smart Contracts

Smart contracts can be described as tamper-proof computer programs [CP22]. It is a special kind of programming interface that can be used to interact with the underlying blockchain storage models [BLS+20]. While Blockchain technology is preferred for data integrity [RBR20] [NNJ19] [AKAK21], Smart contracts are used to provide greater trust, security, efficiency, transparency, and accuracy [Nzu19]. Simplified, they represent digital, self-executing lines of code that are stored and replicated on a blockchain network. They contain a set of predetermined conditions that, when met, automatically trigger the execution of the contract without the involvement of a trusted third-party [B+14]. They can help ensure that policies are always enforced as intended and that fraudulent activity does not occur. The most popular blockchain system for developing smart contracts is Ethereum, which uses the Solidity programming language [sol, WYW+18], and Hyperledger Fabric, which utilizes chaincode typically written in Go [ABB+18]. Batista and Weingaertner specifically conclude in their work that smart contracts have the potential to support some of the record management functions [BW19]. They can be used in various domains, specifically for data management, such as ensuring data

[1] IPFS. https://ipfs.tech/

provenance, data access, or data sharing [KLGG+21]. Particularly, data access is a versatile solution with various applications. Oliveira (2022) and Hao [HHT+22] both focus on attribute-based access control, with Oliveira specifically applying it to medical records sharing and Hao proposing a framework for resource access control. Maesa [dFMMR18] presents a design approach for access control services, leveraging smart contracts on a blockchain and implementing Attribute Managers as smart contracts as well. These studies collectively highlight the potential of smart contracts in various contexts. In the domain of record- keeping, they enable the automation of compliance-related processes by embedding rules and conditions directly into the contract logic. This facilitates the enforcement of defined policies without human intervention and supports auditability through tamper-proof, time-stamped records of all relevant actions.

2.4. Record-keeping with Blockchain Technology

The advent of blockchain has revolutionized various industries by providing a secure, tamper-proof ledger system. Beyond its initial application in cryptocurrencies, blockchain's integration with smart contracts has opened new avenues in sectors like healthcare, supply chain management, voting, and the Internet of Things [XSA+17, XSS+17, CSR+17, BRSS17, KHD17, MSH17, YYNH18, LJW+19, OAEAO16, OEO17, HCK17]. Among these, record-keeping has seen significant enhancements, benefiting from blockchain's inherent properties such as immutability, transparency, and security. Blockchain technology in record-keeping supports crucial functions such as version tracking, tracing document steps within business processes, verification of changes, and streamlined document exchange. This technology assures the integrity of records by confirming their existence at a specific time and their sequence, which is vital for non-repudiation and validation of digitally signed records during long-term preservation. Historically, the first notable application of blockchain in record-keeping was in land registration. Countries like Honduras took pioneering steps, followed by projects in Sweden and Brazil, addressing issues like duplicate titles and unauthorized changes [AMP, CD16, Kei17]. The system in Georgia further exemplifies blockchain's impact, where property ownership information is recorded with enhanced security and transparency [LND19]. Moreover, blockchain applications are explored in healthcare for managing medical records and in finance for maintaining auditable financial records [Lem17b].

Lemieux [Lem16] demonstrates the potential of blockchain as arecord-keeping technology by using international record-keeping and digital preservation standards as a framework for evaluation. These international requirements largely overlap with the requirements for audit-proof archiving, even though the criteria are named slightly differently. In doing so, she identifies that the most important requirement is trust in the digital record. Using a scientific archival theoretical framework, she describes the characteristics that must be in place to achieve this trust. Trustworthiness in records is fundamentally achieved through three interrelated characteristics: accuracy, reliability, and authenticity [Lem17a]. Authenticity describes the quality of the recording, which must be free of manipulation and corruption. While authenticity can be guaranteed by cryptographic encryption of the

data, this does not apply to the reliability and accuracy of the information when it is created off-chain. The truthfulness of the content (accuracy) as well as the competence of the author, the completeness, and the controls during the creation of the digital record (reliability) cannot be verified in the blockchain [DR12].

Weingärnter et al. [WBKV21] is even more specific when it comes to the requirements. He mapped the record-keeping principles directly to the known properties of blockchain technology. According to him, the blockchain in combination with the use of smart contracts is able to fulfil the principles of accountability, transparency, integrity, protection, and availability. However, in a regular blockchain environment, the principles of retention and disposition are not present. These need to be considered in the design of the system through the appropriate use of smart contracts.

The fundamental design of blockchains for permanent storage conflicts with the raditional lifecycle of documents and regulations for proper disposal. This can pose challenges for organizations, as permanent data retention leads to significant storage demands. Since data on the blockchain cannot be deleted, documents should be stored off-chain with only a hash pointer recorded on-chain. This approach allows the document to be deleted off-chain without affecting the integrity of the blockchain [LHBJ19].

Building on the theoretical insights from Lemieux and Weingärnter, practical implementations of blockchain in record-keeping vividly demonstrate the technology's potential. In the implementation of a prototype by the State Committee for Archives of the Republic of Tatarstan, blockchain technology was used as an administrative layer before archiving within an existing archival system. This implementation confirmed that blockchains can effectively be used to ensure data integrity in electronic archives [SB21].

Kallis and Belloum showed that data integrity can be secured by using blockchain based hash validation. By storing the actual data separately and only using the hash value of it on the chain, manipulations and accidental changes can be detected at any time when the actual data is validated against the hash on the blockchain [KB18]. Off-chain storage was used as blockchains are designed to support highly secure, immutable, and transparent transactions, but they are not designed to handle large amounts of data. Storing data on blockchains would create scalability issues as the chain continues to grow [Ben14].

Renner et al. examined the use of blockchain as an auditing framework in record-keeping, where smart contracts are utilized for functions such as file validation, proof of existence, and file history tracking. In addition to ensuring data integrity through document hash values, any file modifications, including timestamps and user information, are recorded on the blockchain. This enables a seamless audit trail and ensures complete traceability and verification of file history [RMK18].

Distinctively, the Lekana project a decentralized document archive storage platform, leverages smart contracts to facilitate complex archival functions such as creation, retrieval, and validation of documents. The approach underscored the versatility of smart contracts in extending blockchain functionalities. Similar to the method of Kallis and Belloum the integrity verification is done by comparing the hashes with the archived document data using smart contracts [BLS+20].

3. Research Question

In the previous section, the requirements for audit-proof archiving were comprehensively described. The GoBD specify and expand the requirements of the Commercial Code (HGB) and the Fiscal Code (AO) through additional regulations necessary for electronic record retention. Furthermore, the IDW RS FAIT 3 supplements these regulations with specific security requirements that must be considered when implementing secure archiving procedures. A literature-based analysis of various implementations of blockchain technology in record-keeping has shown that this technology can offer significant advantages for archiving. The specific properties of blockchain, such as immutability, transparency, data provenance, and the ability to execute programmable logic through smart contracts, suggest that the technology may be suited to support several of the core requirements in audit-proof archiving. Smart contracts, in particular, offer mechanisms for automating rules and policies, potentially enabling greater control, reliability, and traceability in managing digital records. Despite the recognizable advantages and identified potential of blockchain technology, its application in ensuring compliance with the requirements for an audit-proof archiving system has not yet been fully explored. This gap leads to the following central research question of this thesis:

RQ1: To what extent can blockchain technology support the legal requirements of an audit-proof archiving system?

To explore this question in a practical context, this thesis proposes the design and implementation of a proof-of-concept that applies blockchain technology to support key aspects of a compliant audit-proof archiving system. Based on the current state of our research, four sub-questions have been identified that will guide the focus of this work. They are derived from the potentials identified in Chapter 2 in the context of blockchain-based record-keeping and form the basis for the conceptual development of the system:

RQ1.1: How can blockchain technology be used to ensure the integrity of archived data?

RQ1.2: How can blockchain technology be used to support the requirements related to secure access of archived information?

RQ1.3: How can blockchain technology be used to support the requirement for data retention?

RQ1.4: How can blockchain technology be used to support the requirement for traceability and verifiability?

4. Requirement Analysis

This chapter provides a summary of the requirements to be met by an audit-proof archiving system. These requirements are derived from the guidelines outlined in Chapter 2, specifically from the GoBD and IDW RS FAIT 3. The criteria listed therein are categorized into functional and non functional requirements.

4.1. Functional Requirements

First, the functional requirements that the system must fulfil are examined. These requirements define the specific functionalities that need to be implemented within the system.

FR1: Order The system must provide the capability to create a structured electronic filing system that ensures clear and traceable document retention while supporting unique document identification.

FR2: Retention The archiving system must be capable of storing large amounts of data in a complete and unaltered manner for defined retention periods and properly deleting them once the retention period expires.

FR3: Correctness The system must ensure that documents are recorded correctly. Data transmitted to the archiving system must not be falsified or lost.

FR4: Traceability & Verifiability The archiving system must ensure that data storage and all associated processes remain traceable throughout the entire retention period. Additionally, all relevant processes, procedures, and system components must be documented comprehensively in a procedure documentation.

FR5: Timely bookings and recording The archiving of data and documents must be carried out as early as possible to minimize the risk of loss and manipulation.

FR6: Completeness The system must ensure that all data is fully and seamlessly recorded.

FR7: Data Access The system must be capable of providing the tax authorities with all relevant data in a machine-readable format as part of an external audit.

FR8: Machine Readability The system must ensure that all data and documents are machine-readable, allowing for full-text searches, mathematical and technical evaluations, and the tracking of links and references.

4.2. Non-Functional Requirements

The non-functional requirements describe the general characteristics that the system must meet.

NF1: Confidentiality: It must be ensured that data is protected from unauthorized access, such as viewing, disclosure, and publication.

NF2 : Authorization: The system must ensure that only authorized users can exercise their assigned rights and process the archived data and documents.

NF3: Integrity: The system must store the data and documents in a verifiably complete and unaltered form and protect them against manipulation and unintentional or incorrect changes.

NF4: Availability: The system must ensure that the data subject to retention requirements remains readable and retrievable throughout the entire retention period.

NF5: Immutability: The system must ensure that archived data cannot be subsequently suppressed, overwritten, deleted, or falsified without detection until the end of the retention period.

NF6: Data Security: The archive system must be protected against loss and unauthorized inputs and modifications.

NF7: Authenticity: The archived documents must have a clear and verifiable link to the original document and be able to be assigned to a specific business transaction.

NF8: Internal Control System: The system must ensure the proper execution and monitoring of all system processes through organizational regulations and technical measures.

NF9: Binding Character The system must be able to ensure intended legal consequences.

5. Concept

After the requirements for an audit-proof archiving system were derived in Chapter 4, this chapter introduces a fundamental concept for its implementation. The focus lies on the central research question of how blockchain technology can be specifically utilized to support these requirements. First, a necessary delimitation of the scope of this thesis is made before different architectural approaches for a blockchain-based solution are examined. Subsequently, four key functions are introduced through which blockchain, in combination with smart contracts, addresses specific requirements within the archiving system. Finally, a proof of concept (PoC) is described, serving as the foundation for the subsequent implementation.

Delimitation

A complete audit-proof archiving system includes numerous sub-processes. These contain IT-supported processes for data capture, indexing, storage and management, retrieval, and deletion of data and documents. In practice, archiving systems are often integrated with upstream accounting systems for automated data acquisition and downstream storage systems for long-term archiving via defined interfaces [dWFfI06]. As a complete archiving system would have to include all the sub-processes mentioned, its development would go beyond the scope of this thesis. To ensure alignment with the research questions, a targeted delimitation is therefore made, particularly concerning the sub-process of document capture, which includes the transfer, digitization, and classification of both physical and electronic documents. As a result, no automated controls or validations regarding format, origin, completeness, or content correctness are applied to the documents transferred to the archiving system. By excluding the sub-process of document capture, the following requirements cannot be considered:

- Correctness (FR3)
- Timely bookings and recording (FR5)
- Completeness (FR6)
- Authenticity (NF7)

Additionally, the following aspects are not considered:

- The creation of a complete procedural documentation (FR4)
- The possibility of data access for tax authorities (FR7)

- Ensuring intended legal consequences (NF9)
- The machine readability of documents (FR8)
- Protection against data loss (NF6)
- The implementation of an internal control system (NF8)

To ensure that documents are captured correctly, in a timely manner, and completely (FR3, FR5, FR6), the archiving system would need to be equipped with appropriate capture software featuring plausibility and validation controls [dWFfI06]. In practice, this is typically achieved through specialized scanning software or interfaces to accounting systems. These enable the direct transfer of data to the archiving system, where IT controls then verify the correctness and completeness of the data [fwVe]. Since a detailed implementation of the capture process falls outside the focus of this thesis, these requirements are not taken into account for the system to be developed.

The requirement for Authenticity (NF7) mandates that archived documents must have a unique and verifiable connection to the original document and be assignable to a specific business transaction. However, establishing such a link would require access to relevant booking and transaction data from upstream ERP or accounting systems. It is important to note that the definition of authenticity in archival science, referring to the genuineness and integrity of information, can indeed be ensured using blockchain technology [Lem]. However, this does not correspond to the requirement in this context. Furthermore, a complete procedural documentation is not created as part of the requirement for traceability and verifiability (FR4), as a detailed description of all relevant processes, procedures, and system components is not relevant to the central research questions. However, since this requirement also concerns the traceability and verifiability of data and documents within the system, it is partially considered. Additionally, the possibility of data access for the financial administration (FR7) is excluded from consideration. The necessary setup for external access (Z1, Z2, Z3) or data export would exceed the scope of this thesis and is therefore not addressed. This thesis focuses on assessing the extent to which blockchain technology can meet legal requirements for audit-proof archiving. However, it does not examine the specific legal implications of using blockchain technology. Whether and to what extent the system can ensure intended legal consequences in accordance with the requirement of binding character (NF9) is beyond the scope of this thesis. Moreover, machine readability of data such as full-text searches, mathematical-technical evaluations, and tracking of links and connections is not supported due to the complexity of its implementation. The GoBD specifies measures for protecting the archive system from data loss as part of the requirement for data security (NF6). However, implementing a comprehensive backup and recovery concept would also exceed the intended scope of this thesis. Such measures require detailed planning, redundant storage systems, and contingency strategies, which could be explored in future research. Although certain control mechanisms within the archive system are considered, a complete implementation of an Internal Control System

(NF8) requires several preventive and detective controls that are conducted within the subprocess of data capture. Since, as previously mentioned, the process of data capture is not a focus of this thesis, the implementation of a comprehensive Internal Control System is omitted. Table 5.1 provides a comprehensive overview of the considered and delimited requirements:

Table 5.1.: Requirement Overview

Type	**Requirement**	**Consideration**
Functional	FR1: Order	✓
	FR2: Retention	✓
	FR3: Correctness	✗
	FR4: Traceability & Verifiability	(✓)
	FR5: Binding Character	✗
	FR6: Timely bookings and recording	✗
	FR7: Completeness	✗
	FR8: Data Access	✗
	FR9: Machine Readability	✗
Non-Functional	NF1: Confidentiality	✓
	NF2: Authorization	✓
	NF3: Integrity	✓
	NF4: Availability	✓
	NF5: Immutability	✓
	NF6: Data Security	✗
	NF7: Authenticity	✗
	NF8: Internal Control System	✗

✓ indicates that the requirement is considered in this thesis. (✓) indicates a partial consideration of the requirement. ✗ indicates the delimited requirements.

5.1. Architectural Design

Various architectural designs suitable for the system were considered during the exploration. In this context, the blockchain can either be implemented as a decentralized application

(dApp) or in a hybrid form within larger enterprise solutions, where centralized components are selectively integrated. Figure 5.1 illustrates the classic 3- tier application design in three variations: traditional, decentralized, and hybrid. Since the developed application is intended to integrate blockchain in any case, the traditional variant is not considered. Developing a decentralized application that relies exclusively on distributed components often poses challenges due to technical limitations and usability issues. In this context, [WZRM21] describes the constraints of smart contracts in interacting with off-chain services outside the closed blockchain ecosystem to retrieve information or trigger actions.

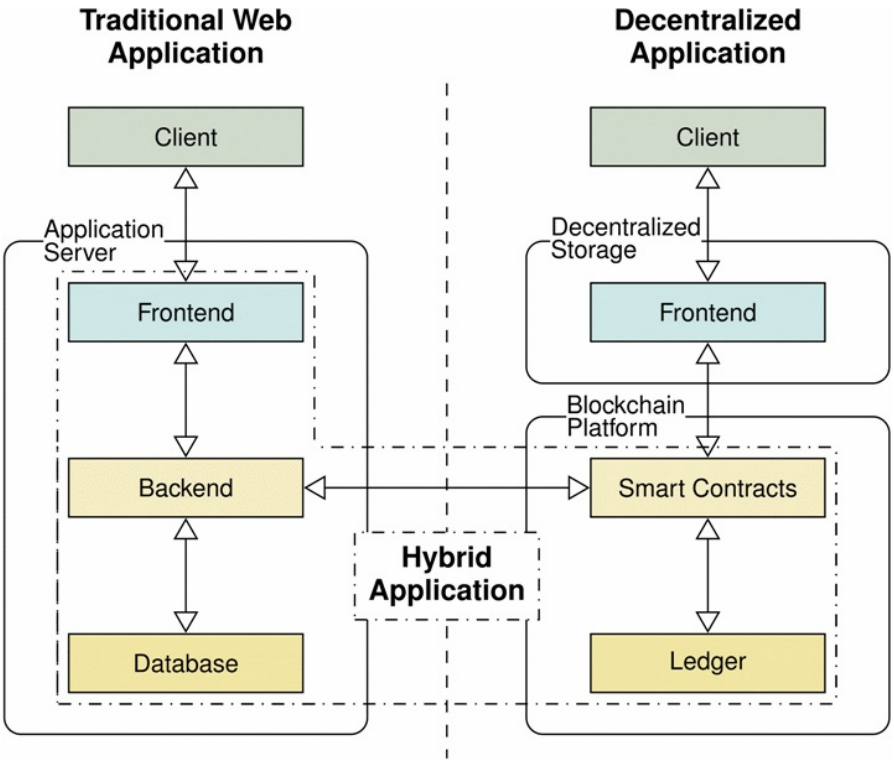

Figure 5.1.: A traditional, hybrid, and decentralized application architecture [WZRM21]

For this reason, the concept follows a hybrid (semi-decentralized) approach in which the server integrates external data and handles routine tasks such as archiving documents in an off-chain storage system [WZRM21]. In this approach, the blockchain serves as a supporting tool in the form of an integrity layer, while smart contracts can be used to support the archiving processes.

5.2. The Role of Blockchain in the Archiving System

In Chapter 2, it was shown that blockchain technology has several inherent properties that can support audit-proof archiving. In particular, immutability and traceability are

key features where the technology can provide added value for archiving. By using smart contracts, additional policies defined as requirements in Chapter 4 can be automated. The concept is presented below.

- **Storing cryptographic hash values on-chain**
  Based on the method described in Chapter 2 by Kalis and Belloum, a unique hash value is generated for each document and stored on the blockchain. By comparing the stored hash value with a newly computed hash of the document, it is possible to verify at any time that the data has remained complete and unaltered throughout the entire retention period. This approach addresses the requirement for integrity **(NF3)**. The inherent property of immutability **(NF5)** ensures that once data is written to the blockchain, it can no longer be modified. Any discrepancy between the reference value stored on the blockchain and the data in the archival storage indicates potential document tampering [KB18].

- **Verifying and managing access rights to archival data**
  The IDW RS FAIT 3 mentions the use of access controls as a way of supporting the confidentiality requirement **(NF1)** by protecting the data from unauthorized access [dWFfI06]. The use of smart contracts for access control was emphasized in the state of the art 2.3.5. Conventional access control models have the disadvantage that they rely on centralized and trusted servers that mediate every access attempt. This can lead to problems such as a single point of failure and a lack of transparency [HHT+22]. By verifying permissions through Smart Contracts, access control becomes tamper-proof and transparent. Smart Contracts automate the authorization process, making it immune to manipulation, which can enhance both security and efficiency. Therefore, in the system to be developed, a simple management of access rights will be implemented on the blockchain using Smart Contracts. In accordance with the authorization requirement **(NF2)**, the implementation aims to ensure that only authorized users can access the archival data. Authorization includes the assignment of rights within the smart contract so that users can only perform the actions for which they are authorized.

- **Enforcing retention periods**
  Chapter 2.3.4 indicates that blockchain technology is not primarily suited for storing large volumes of data, and it is therefore recommended to use an off-chain storage solution 2.4. In audit-proof archiving, it is essential to store documents according to prescribed retention periods. Smart Contracts offer a more efficient solution in this context, as they can autonomously enforce data retention rules, remain immutable, and ensure a consistent application of legal requirements. In the system to be developed, Smart Contracts will be used to ensure compliance with retention periods. This is intended to fulfill part of the requirements for data retention **(FR2)**. Within the Smart Contract, not only the hash values of the archived documents will be stored, but also the associated metadata, including retention periods. This allows for an automatic verification of whether the deletion of a document complies with

legal requirements, preventing premature removal of documents from the archival system.

- **Immutable logging of document-related interactions**
  The use of blockchain as an auditing framework in record-keeping has also been mentioned in chapter 2. By immutably documenting all modifications on the blockchain, the traceability of the file history is ensured 2.4. In a GoBD checklist published by the German Federal Association for Information Technology, Telecommunications, and New Media (Bitkom), logging functions are specifically highlighted as an implementation option to support the requirement of traceability and verifiability **(FR4)**. These functions contribute to demonstrating the proper operation of the system and ensure a detailed file history of each object in the archive. The log should include the action, date time, and the person involved, whereby the logs must be protected against loss and tampering throughout the entire retention period [BI]. Therefore, the system to be developed should support the logging of all user actions related to archived data objects within a smart contract. To make logging more comprehensible and accessible, Smart Contract Events are used, functioning as readable logs that record the execution of important actions [goe]. They enable a verifiable documentation of interactions with the archived data object. This approach is designed to make it also easier for auditors to clearly check and verify transactions and processes in the system. In this concept, these processes include not only uploading, accessing, and deleting documents but also managing access permissions. Since the logs are linked to blockchain transactions, they remain permanently and immutably stored, providing a significant advantage over traditional system logs.

5.3. Proof of Concept

To examine the practical application of blockchain technology for the functionalities described in Chapter 5.2 within the archiving system, a prototype will be developed. Since the system uses a hybrid architecture, it will include both blockchain technology and traditional IT components.

- Blockchain (Ledger)
- Smart Contract
- Off-chain Storage (Database)
- Backend
- Frontend

In the designed system, the blockchain serves as a fundamental integrity layer that supports key archiving processes through smart contracts. Users can upload and securely

store documents, verify their integrity throughout the entire retention period, access archived data (if authorized), view a transparent history of all interactions with the data objects, and delete documents in compliance with regulations once the retention period expires. The blockchain acts as an immutable and transparent source of proof, while smart contracts support integrity enforcement, access control, compliance with retention periods, and logging.

The backend serves as the connecting component of the architecture, linking the blockchain and smart contracts with the off-chain storage and frontend. It processes user requests, initiates blockchain transactions, and manages communication with the off-chain database. Storing data directly on the blockchain is computationally intensive and time-consuming [ITL+22]. As recommended in research, documents are therefore stored in encrypted form in an off-chain storage, while only hash values of the corresponding documents and their metadata are stored on-chain. This solution allows for efficient storage and management of large data volumes without compromising blockchain performance [GBD23, DYA+23]. Finally, a frontend enables users to interact directly with the system. Through the frontend, users can upload documents, retrieve them if authorized, and manage access rights to archived data. Additionally, users can view the log history of archived data and initiate compliant document deletion after the retention period has expired. The PoC implements additional functionalities in the traditional components of the system in order to meet the requirements for audit-proof archiving. These include unique document identification to support the requirement for order, the provision of archive data via a web interface to support the requirement for availability, and the storage of documents to support the requirement for retention. These are described in chapter 7 as part of the respective system component. However, as these don't include the use of blockchain technology, they are to be understood as a supplement to be able to test the blockchain supported functions in a practical scenario.

6. Technology Stack

In this chapter, the technologies used for implementing the system will be described. Initially, an appropriate choice for the blockchain platform and the associated off-chain storage needs to be made. Subsequently, the most important dependencies utilized within the system will be explained.

6.1. Discussion on the choice of Blockchain

When selecting a blockchain for enterprise use, two of the most prominent platforms, Hyperledger Fabric and Ethereum, are often compared [DKTA20].

Hyperledger Fabric

Hyperledger Fabric[1], as a permissioned blockchain platform, inherently provides strong control over data privacy and access rights. Features such as private channels and the ability to restrict transaction visibility to specific network participants make Fabric seem ideal for implementation at first glance. Additionally, Fabric's modular architecture allows for a high degree of adaptability to specific business needs and compliance requirements. However, the platform requires complex, multi-level configuration, including the setup of certificate authorities, creation of channels, deployment of chaincode, and the establishment of peers and orderers. This can be particularly challenging for users already familiar with other blockchain systems like Bitcoin or Ethereum. Furthermore, Fabric supports smart contracts that can be written in various programming languages, but these languages are not specifically optimized for blockchain applications, which could introduce further challenges [YNZ+19].

Ethereum

With the transition to the Proof-of-Stake (PoS) consensus algorithm on September 15, 2022, some of the existing drawbacks of the Ethereum blockchain[2] have been addressed. Compared to the former Proof-of-Work (PoW) algorithm, this change has significantly improved sustainability by reducing energy consumption by approximately 99.95%. Furthermore, the transition aimed to achieve greater decentralization. Ethereum 2.0 requires a minimum number of 16,384 validators, making the network significantly more decentralized and thus potentially more secure compared to other PoS networks, which often have a smaller number of validators and therefore represent more centralized and potentially

[1] Hyperledger Fabric. https://www.hyperledger.org/projects/fabric
[2] Ethereum. https://ethereum.org/

less secure systems. The introduction of sharding represents another significant upgrade that will greatly enhance Ethereum's scalability and performance in the upcoming years. This method promises to improve network efficiency by dividing the network into multiple smaller parts, enabling transactions to be processed in parallel, as only a subset of validators are responsible for a specific group of transactions [ethb]. Ethereum's continued development, particularly through the transition to PoS and the implementation of sharding, makes the platform more sustainable and performant. However, using a public blockchain in an enterprise context presents specific challenges. While public, unrestricted blockchains provide high transparency and data integrity, they can raise concerns regarding data privacy and security. Here, private, permissioned blockchains, such as Hyperledger Fabric, offer advantages by enabling more precise control over participants and blockchain activities, including data protection and access controls. Additionally, such systems do not require cryptocurrencies, which can represent a hurdle for many enterprises

The specific blockchains raise questions regarding their suitability for the implementation of the PoC, particularly concerning data privacy (Ethereum) and implementation complexity (Hyperledger). By including the Ethereum client GoQuorum[3] into the discussion, some of the limitations can be addressed.

GoQuorum

GoQuorum is an Ethereum-based client specifically developed by J.P. Morgan for enterprise applications. As a fork of Ethereum, GoQuorum retains many fundamental Ethereum features but extends them with additional functionalities optimized for private networks and applications requiring a high degree of data privacy. Among its key features is the processing of private transactions, in which transaction details are only visible to involved participants, while the transaction itself remains verifiable on the blockchain. Furthermore, GoQuorum supports multiple consensus algorithms such as Raft[4], Clique[5], and Istanbul Byzantine Fault Tolerance (IBFT)[6], enabling flexible network configurations and improved performance tailored to specific use cases. Similar to Hyperledger, GoQuorum offers high scalability with a diverse selection of consensus mechanisms, allowing adaptation to particular application scenarios. Each of these mechanisms has its own distinct characteristics regarding speed, fault tolerance, and finality, making GoQuorum adaptable to various enterprise requirements.

For the implementation of the concept, the final choice fell on the Ethereum-based client GoQuorum. GoQuorum stands out due to several key characteristics, making it particularly suitable for the system being developed. These include its variety of consensus algorithms and high scalability. The command-line tool QuorumWizard significantly simplifies the setup of an initial development network and offers services to support network monitoring. This greatly reduces the complexity and effort involved in building

[3] GoQuorum. https://docs.goquorum.consensys.io/
[4] Raft. https://raft.github.io/
[5] Clique. https://github.com/Consensys/quorum/tree/master/consensus/clique
[6] IBFT. https://goquorum.readthedocs.io/Consensus/ibft/ibft/

and managing the blockchain infrastructure. Compatibility with Ethereum allows the utilization of established development tools and resources from the Ethereum community, such as the JSON-RPC API[7], as well as extensive documentation. Furthermore, support for the Ethereum Virtual Machine (EVM) enables the use of the specialized smart contract programming language Solidity. Particularly relevant in an enterprise context are GoQuorum's additional privacy and security features, which enable private transactions, thus ensuring the confidentiality of critical data stored on the blockchain. Although the configuration of the system implemented within this work does not actively utilize these features, the option remains open for future adaptations and extensions.

6.2. Discussion on the choice of Off-Chain Storage

For storing large amounts of data, the system has to employ an off-chain storage solution. In Chapter 2.3.4, several possible approaches were named. Due to the ongoing trend among companies towards outsourcing increasing amounts of data to the cloud, traditional databases hosted on corporate servers will not be considered further. Among decentralized storage systems, Filecoin[8] and IPFS[9] particularly stand out in blockchain-based applications. These systems provide alternatives to conventional databases by distributing the storage across a broader range of network participants, thereby potentially enhancing fault tolerance. With recent changes to the GoBD in 2020, it is now also possible to use cloud systems for processing and storing company documents. Given the increased relevance due to these legal adjustments and the overall trend towards cloud adoption, the centralized cloud storage solution Google Cloud Storage will also be included in this analysis.

IPFS

As a decentralized storage system, IPFS offers advantages by distributing data across a peer-to-peer network. Although this generally ensures high data availability, IPFS does not guarantee long-term persistence, as it lacks a direct incentive mechanism for permanent data storage [DT22, DPOB22]. This conflicts with the legal requirements of the GoBD, which mandate secure and long-term retention of corporate data. Additionally, ensuring that data remains stored domestically within a peer-to-peer network is difficult to control. According to § 146 subsection 2b sentence 1 of the German Fiscal Code, each storage node would require individual approval from the tax authorities, resulting in an impractical amount of effort. Finally, IPFS often lacks essential data protection mechanisms [DPOB22], thereby conflicting with compliance regarding confidentiality and security requirements.

[7] JSON-RPC API. https://ethereum.org/en/developers/docs/apis/json-rpc/
[8] Filecoin. https://filecoin.io/
[9] IPFS. https://ipfs.tech/

Filecoin

Filecoin addresses some of the limitations inherent to IPFS by implementing an incentive mechanism that rewards long-term data storage [IPF]. Data is transmitted and stored in encrypted form, while miners regularly provide cryptographic proofs of proper storage. These mechanisms facilitate compliance with availability requirements. Contracts with miners include terms specifying storage duration, further supporting adherence to retention obligations. However, since Filecoin operates as an incentive layer on top of IPFS, it faces the same challenges concerning the storage location according to § 146 subsection 2b sentence 1 of the German Fiscal Code. Furthermore, the use of cryptocurrencies for transactions within the Filecoin network could represent an additional obstacle for enterprises.

Google Cloud Storage

Google Cloud Storage[10] (GCS), as a modern cloud storage solution, offers a wide range of storage options that can be adapted based on access frequency and cost efficiency. Of particular relevance is the "Archive Storage" class, explicitly designed for long-term data archiving. Data storage locations can be configured in various regions, including Germany, thus satisfying the location requirement for data retention. Using multi-regional buckets, data availability can also be ensured in disaster recovery scenarios, as data is replicated across multiple locations simultaneously, further enhancing system resilience.

While decentralized storage solutions could represent an interesting alternative, they currently do not meet the stringent requirements for audit-proof archiving. The decision to adopt a cloud storage solution was influenced by the ongoing trend toward moving data into the cloud to achieve greater flexibility, scalability, and cost efficiency. Established cloud solutions such as Google Cloud Storage not only fulfill legal obligations concerning data availability and retention but are also internationally certified according to various standards. For the PoC, the selection of Google Cloud Storage serves only as an example implementation to demonstrate the overall system's potential. Other cloud storage providers could also be considered, provided they offer comparable functionalities and meet necessary compliance criteria. However, to meet the confidentiality requirements **(NF1)** in the proof of concept, it must be ensured that data is properly encrypted before being stored.

6.3. Dependencies

[10] Cloud Storage. https://cloud.google.com/storage

Solc

The Solidity Compiler[11] (Solc) is an essential tool for developing smart contracts for Ethereum blockchain-based applications. This tool translates code written in Solidity, a programming language specifically designed for smart contracts, into bytecode. This bytecode is crucial for execution on the Ethereum Virtual Machine (EVM). In addition to generating the bytecode required for deploying the contract on the Ethereum blockchain, the compiler also produces the Contract Abstract Binary Interface (ABI). The ABI is a critical component enabling interaction between the smart contract and other applications, as it functions as an interface ensuring the correct and secure execution of smart contract functions.

Go-Ethereum

Go-Ethereum[12] (Geth) is an official implementation of the Ethereum protocol in Go and serves as an essential library for interacting with the Ethereum blockchain. It enables the management of Ethereum nodes, the sending of transactions, and access to blockchain data via JSON-RPC interfaces. Additionally, it provides an API for integrating smart contracts into Go applications and supports fundamental blockchain functions.

Web3

Web3.js[13] is a JavaScript library that facilitates direct interaction with the Ethereum blockchain and its smart contracts within web applications. By utilizing Web3.js, blockchain functionalities can be seamlessly integrated into user interfaces, significantly simplifying and accelerating the development of blockchain-based applications.

PostgreSQL

PostgreSQL[14] is an advanced open-source object-relational database management system (ORDBMS) known for its reliability, robustness, and performance. It offers comprehensive features for transaction integrity, data security, and support for complex queries and data models. Its scalability and efficient handling of large datasets make it particularly suitable for applications with high interaction rates and complex data relationships.

Programming Language - Backend

For the backend programming language of this project, GoLang[15] was chosen, primarily due to its seamless integration with go-ethereum (Geth), the official Go implementation of the Ethereum protocol. This integration provides an efficient development environment

[11] Solc. https://docs.soliditylang.org/en/latest/installing-solidity.html
[12] Go-Ethereum. https://geth.ethereum.org
[13] Web3.js. https://web3js.readthedocs.io/en/v1.10.0/
[14] PostgreSQL. https://www.postgresql.org/
[15] Go. https://go.dev/

for Ethereum-based applications and smart contracts. Utilizing the go-ethereum (Geth) library grants access to a comprehensive set of tools and functions specifically designed for the Ethereum blockchain. Additionally, support from an active GoLang and Ethereum developer community offers a valuable advantage, ensuring access to extensive resources and expertise.

Programming Language - Frontend

JavaScript[16] is a preferred language for web development, offering a flexible and powerful environment for creating dynamic and interactive user interfaces. Broad support from modern frameworks and libraries like React.js enables the development of single-page applications (SPAs) and component-based architectures, ensuring a seamless and efficient user experience.

React

React[17] is a JavaScript library that facilitates the creation of interactive and dynamic web pages by structuring user interfaces into independent units known as components. This approach allows developers to update parts of a webpage without requiring a full reload. Each component in React is independent and manages its own state, meaning only the modified sections of the page are re-rendered. This significantly enhances user experience by improving the speed and responsiveness of applications. Vite[18] is a modern frontend development tool that supports this process by reducing development and build times. It offers quick setup and immediate feedback through fast hot-reloading when code changes. By directly integrating with React, Vite simplifies development by eliminating the need for complex configurations, allowing developers to focus on actual application development.

Docker

Docker[19] enables the containerization of the entire application, ensuring a simplified and efficient setup and management of both development and production environments. ConsenSys provides a Docker Compose configuration for GoQuorum, which deploys and manages a private IBFT network of GoQuorum nodes. This setup also includes supporting containers for monitoring, such as Prometheus and Grafana. Utilizing Docker ensures that all application components, such as the frontend, backend, and blockchain, run in a consistent and isolated environment, significantly reducing system management complexity.

[16] JavaScript. https://developer.mozilla.org/en-US/docs/Web/JavaScript
[17] React. https://react.dev/
[18] Vite. https://vitejs.dev/
[19] Docker. https://www.docker.com/

7. Implementation

This chapter provides a detailed description of the system's implementation. Each component is examined separately, explaining how the specific requirements from Chapter 4 have been implemented within the respective system components. Figure 7.1 first presents a schematic overview of the system architecture.

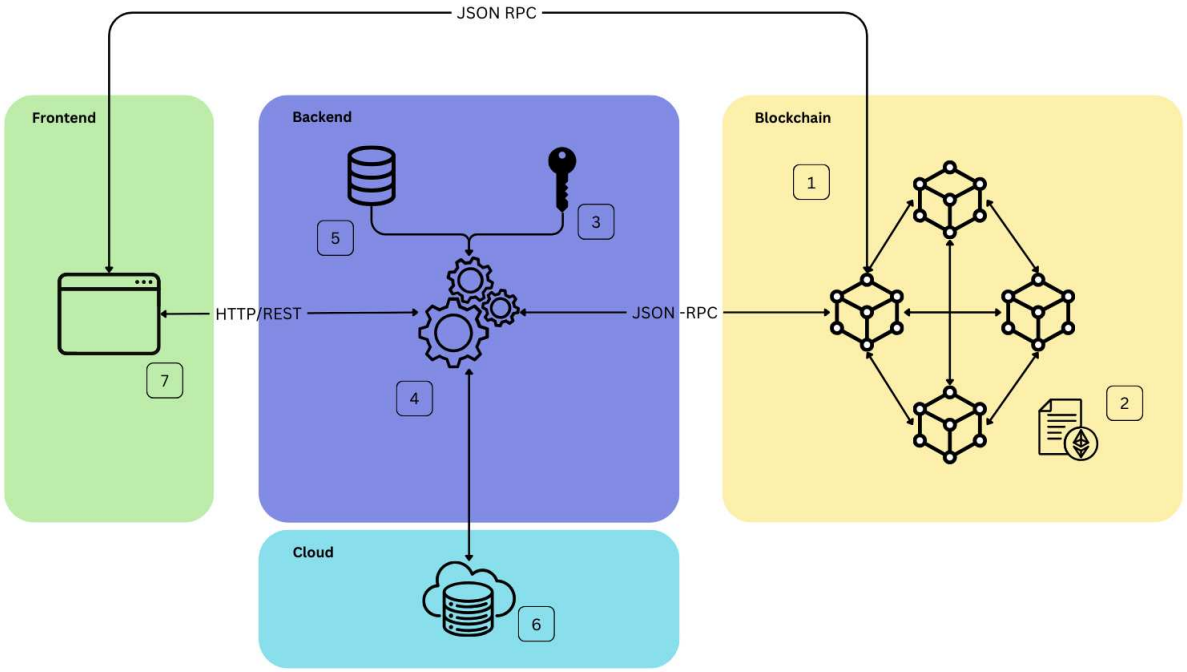

Figure 7.1.: Architecture of the designed System
-based on [WZRM21]

7.1. Blockchain

In the designed system, the blockchain **(1)** is used as an additional integrity layer. It not only enhances data integrity but also strengthens user confidence in the authenticity and verifiability of stored information. Through the implementation of the blockchain, the following requirements are supported:

- Availability (NF4)
- Immutability (NF5)

Looking at the blockchain individually, it supports the system with two inherent properties. Firstly, due to the cryptographic chaining of blocks, data that has been recorded on the blockchain can no longer be altered or manipulated [HWRBS17]. Secondly, the blockchain increases the availability of the stored data through its decentralized structure [WBKV21]. The blockchain stores the hash values of documents and their metadata as a digital fingerprint, ensuring their immutability in accordance with requirement **NF5**. This provides verifiable proof of the document's state at the time of archiving and enables the detection of any subsequent modifications to archived data objects. At the same time, each participating node in the blockchain network maintains a redundant copy of the stored data, enhancing its overall availability **(NF4)**. However, this availability only applies to on-chain data, so additional measures are needed to ensure the requirement.

7.1.1. Network Setup

The network implements the Quorum Byzantine Fault Tolerance (QBFT) consensus algorithm, which is specifically designed for enterprise use in private blockchains. QBFT requires the approval of four validators to achieve consensus and ensure fault tolerance. These validators can represent different organizations or different divisions, or branches of the same company. For the proof of concept (PoC), the implementation is designed for a single company with multiple branches as users. Each additional peer node must be verified by the blockchain operators, ensuring that only known and trusted organizations can participate in the network. The initial configuration of the genesis block was adjusted in two key aspects to set up the network. First, the parameter "gasLimit" was set to a maximum value, effectively making the network a "Free Gas Network". This means that transactions do not incur costs, which is particularly beneficial for enterprise use. Second, the parameter "blockPeriodSeconds" was set to two seconds. This defines the time interval between block creations, enabling faster transaction confirmations and improving user experience. However, a shorter block time also increases network traffic. For real-world deployment, the optimal block time needs to be carefully evaluated. The implementation configures a Quorum blockchain network using Docker Compose, consisting of multiple services. In addition to the four validator nodes that collectively validate transactions and manage the blockchain, an RPC node is deployed along with additional support services for monitoring and data visualization.

Validator nodes The validator nodes process transactions and generate blocks. Each validator is configured with specific ports that enable HTTP RPC communication.

- validator1, port: 21001:8545
- validator2, port: 21002:8545

- validator3, port: 21003:8545
- validator4, port: 21004:8545

RPC Node The RPC node (rpcnode) serves as the HTTP interface for the backend and frontend to access the network.

- rpcnode, port: 8545:8545

In the standard configuration of GoQuorum, additional services are integrated for monitoring and analyzing the network. Since these are not the primary focus of this work, they are only briefly mentioned.

Quorum Explorer A graphical user interface is available on port 25000. It enables users to track network activities such as transaction history and block analysis.

- explorer, port: 25000:25000

Prometheus Provides a collection of metrics that offer important insights into network performance.

- prometheus, port: 9090:9090

Grafana Uses the data collected by Prometheus to display it in a user-friendly and visually appealing format.

- grafana, port: 3000:3000

7.1.2. Smart Contract

With the initial start of the application, a smart contract is automatically deployed on the blockchain. The DocumentStorage smart contract **(2)** is written in Solidity and provides fundamental functions to support the storage and management of documents.
The following requirements are supported:

- Compliance with retention periods (FR2)
- Comprehensive logging of archiving operations (FR4)
- Protection against unauthorized access (NF1)
- Management of access rights (NF2)
- Ensuring the integrity of data and documents (NF3)

An overview of the data stored in the smart contract for an archived document is presented in Table 7.1. All documents are centrally organized using a mapping structure, where a UUID (Universally Unique Identifier) serves as the identifier for each document. Each document is linked to a file hash and a metadata hash, both stored on-chain to verify the integrity of the data and documents. For hash generation, the SHA-256 hashing function was chosen, as recommended by the technical Guideline BSI TR-02102-1 "Cryptographic Mechanisms: Recommendations and Key Lengths" [fSidI24]. According to current knowledge, SHA-256 is considered cryptographically secure. Additionally, the data structure contains information about the document uploader, the retention period, and access permissions. The active/inactive status of a document addresses the challenge of data disposition on the blockchain. Since data cannot be deleted from a blockchain, this status is used to indicate whether a document has been removed from the off-chain storage.

Table 7.1.: Data Structure

Element	**Description**
uuid	Unique identifier of the document
fileHash	Cryptographic hash of the document
metadataHash	Hash of the metadata, including information such as filename, size, and MIME type
retentionDate	Date until which the document must be retained
uploader	Address of the user who uploaded the document
hasAccess	Mapping of which users have access to the document
accessList	List of all users granted access
isActive	Status of the document (active or deleted)

Each write operation in the smart contract represents a state change, which, as described in Chapter 5.2, is specifically logged using events. These events play a crucial role in ensuring transaction transparency and maintaining a traceable history of all significant actions within the contract [goe]. They provide details about who performed a given action, what was done, and when it occurred. By indexing selected parameters, events become easier to search and filter, which significantly enhances the ability to identify specific transactions. This mechanism directly supports the requirement for comprehensive logging as outlined in FR4 (Traceability Verifiability). An overview of the recorded events is presented in Table 7.2. To retrieve the logs, emitted events are filtered based on the respective archived data object, allowing all interactions to be transparently traced.

The functions implemented in the smart contract are described below.

Table 7.2.: Events

Event	Description
DocumentUploaded	Logs the upload of a new document
DocumentVerified	Logs the successful verification of a document's integrity
DocumentDeleted	Logs the deletion of a document
DocumentAccessed	Logs the retrieval of a document by a specific user
AccessGranted	Logs the granting of access rights
AccessRevoked	Logs the revocation of access rights

uploadDocument

The smart contract function "uploadDocument" enables the upload of a document and a metadata hash value. When calling this function, a unique identifier (UUID), hash values for the file and metadata, and the retention period must be provided. The function first checks whether the document already exists or has been marked as inactive. If this is not the case, the hash values, along with the additional metadata and UUID, are stored in the smart contract, and the uploading user automatically receives access rights to the specific document. Additionally, verification of document hash values is performed via the "verifyDocument" function. This verification ensures that the archived object matches the document as it existed at the time of upload. If the verification fails, the transaction is reverted, which means that all associated state changes and any emitted events are discarded. If the operation is successful, the *UploadDocument* event is emitted.

```
1
2       function uploadDocument(
3           string calldata uuid,
4           string calldata _fileHash,
5           string calldata _metadataHash,
6           string calldata _fileHashToVerify,
7           uint256 _retentionDate
8       ) external {
9           require(documents[uuid].uploader == address(0) || !documents[uuid
        ].isActive, "Document already exists");
10
11          Document storage doc = documents[uuid];
12          doc.fileHash = _fileHash;
13          doc.metadataHash = _metadataHash;
14          doc.retentionDate = _retentionDate;
15          doc.uploader = msg.sender;
16          doc.isActive = true;
17          doc.hasAccess[msg.sender] = true;
18          if (documents[uuid].accessList.length == 0) {
19              doc.accessList.push(msg.sender);
20          }
21
22          if (!verifyDocument(uuid, _fileHashToVerify)) {
```

```
23                 revert("Hash mismatch - integrity check failed");
24             }
25
26
27             emit DocumentUploaded(uuid, msg.sender);
28
29         }
30         ...
31 }
```

Listing 7.1: uploadDocument

verifyDocument

The "verifyDocument" function ensures the integrity of a document during the upload process by verifying that the archived document matches the original at the time of submission to the archiving system. This is done by generating a SHA-256 hash of the file in the frontend before the upload. The generated hash is then compared with the hash (_fileHash) created in the backend. If the values match, the _fileHash is stored on the blockchain, proving that the data remained unchanged from submission to archiving at a specific point in time. This function ensures data integrity **(NF3)** throughout the retention period, as any modifications to the document would result in a different hash value. The integrity check is automatically performed before every download, and a manual verification can also be triggered at any time during the retention period. If the verification is successful, the operation is logged through the *DocumentVerified* event.

```
1
2     function verifyDocument(string calldata uuid, string calldata
      fileHashToCheck) public returns (bool) {
3         require(documents[uuid].isActive, "Document is not active");
4         bool hashMatch = keccak256(abi.encodePacked(documents[uuid].
      fileHash)) == keccak256(abi.encodePacked(fileHashToCheck));
5
6         emit DocumentVerified(uuid, msg.sender, hashMatch);
7
8         return hashMatch;
9     }
10
11    ...
12 }
```

Listing 7.2: verifyDocument

checkAndDeleteIfExpired

To comply with the retention requirement **(FR2)**, it must be ensured that no archived data objects are deleted before the expiration of the defined retention periods. In the Proof of Concept, this check is implemented through the Smart Contract function "checkAndDeleteIfExpired". This function verifies the retention period that was immutably stored on

the blockchain at the time of document upload, and permits deletion only once that period has elapsed. Implementing this via the Smart Contract aims to prevent unauthorized deletion and enhance the transparency of processes. Additionally, it reduces the risk of human error and information loss, thereby potentially strengthening trust in the integrity of the data and the entire system. With this function, documents can only be marked as inactive after the designated retention period has elapsed. Once the retention period has been reached, the function internally triggers "deleteDocument", which is executed only if the user holds the appropriate access rights to the document. In the PoC, a simple permission scheme is utilized, granting only the uploader the ability to mark a document as deleted. It is crucial to understand that actual deletion of data on the blockchain is impossible. Instead, the smart contract updates the document's status to inactive, preventing further queries from retrieving it from off-chain storage. Upon successful execution, the operation is logged accordingly via the Smart Contract using the *DocumentDeleted* event. Although it would be technically possible to automate deletion upon expiration of the retention period, the GoBD guidelines published by Bitkom explicitly recommend against doing so [BI].

```
1
2     function checkAndDeleteIfExpired(string calldata uuid)
3     external {
4         require(documents[uuid].isActive, "Document already inactive");
5         Document storage doc = documents[uuid];
6         if (block.timestamp >= doc.retentionDate) {
7             deleteDocument(uuid);
8         } else {
9             revert("Retention period not met.");
10        }
11    }
12
13
14    function deleteDocument(string memory uuid)
15    internal {
16        require(documents[uuid].uploader == msg.sender,
17        "Only the uploader can delete the document");
18
19        Document storage doc = documents[uuid];
20        doc.fileHash = "";
21        doc.metadataHash = "";
22        doc.retentionDate = 0;
23        doc.uploader = address(0);
24        doc.isActive = false; // Mark the document as inactive
25
26        for (uint i = 0; i < doc.accessList.length; i++) {
27            doc.hasAccess[doc.accessList[i]] = false;
28        }
29        delete documents[uuid].accessList;
30
31        emit DocumentDeleted(uuid, msg.sender);
32    }
```

```
33      ....
34  }
```

Listing 7.3: deleteDocument

getDocument

The "getDocument" function enables users to retrieve documents based on a unique identifier (UUID). The function first checks whether the document exists and is active. If this condition is met, it further verifies whether the requesting user has the necessary access rights. Once both conditions are met, the system uses the "verifyDocument" function to compare the file hash retrieved from off-chain storage with the corresponding value stored on the blockchain, ensuring data integrity. If the values match, the document is made available to the user. The system logs each retrieval via the *DocumentAccessed* event, capturing the user, the accessed document, and the time of retrieval.

```
1
2
3       function getDocument(string calldata uuid, string calldata
        fileHashToVerify) external {
4           require(documents[uuid].isActive, "Document has been deleted or
        does not exist");
5           require(hasAccess(uuid, msg.sender), "You do not have access to
        this document");
6
7
8           if (!verifyDocument(uuid, fileHashToVerify)) {
9               revert("Hash mismatch - integrity & authenticity check failed
        ");
10          }
11
12          emit DocumentAccessed(uuid, msg.sender);
13      }
14      ...
15  }
```

Listing 7.4: getDocument

Access

A simple access control model was implemented in the smart contract through the use of four functions. The function "hasAccess" allows checking a user's access rights for a specific document. Access permissions are managed using the mapping *hasAccess*, which associates each document with the users that are authorized to access it. When a document is uploaded, the uploader automatically receives access rights and is added to the document's *accessList*. This ensures that the uploader has full control over the document from the beginning and can manage the permissions of other users. Additionally, an *accessList* keeps track of all addresses that have been granted access. The

functions "grantAccess" and "revokeAccess" are essential for managing permissions. With "grantAccess", the uploader can grant another user access to a specific document. Before doing so, the function checks whether the document is active, confirms that the caller is the original uploader, and ensures that the recipient does not already have access. If all conditions are satisfied, the access permission is updated in the *hasAccess* mapping, and the recipient's address is added to the *accessList*. Conversely, "revokeAccess" allows the uploader to revoke a user's access. Similar checks ensure that only the uploader can perform this action and that the user currently has access to the document. These functions form the basis for both authorization and confidentiality, as they ensure that only explicitly authorized users can exercise their assigned rights to process archived documents **(NF2)**, while also protecting the data from unauthorized access **(NF1)**. To maintain traceability, any changes in permissions are documented and properly logged using the events *AccessGranted* and *AccessRevoked*.

```
1
2       function grantAccess(string calldata uuid, address grantee)
3       external {
4           require(documents[uuid].isActive,
5           "Document is inactive or does not exist");
6           require(documents[uuid].uploader == msg.sender,
7           "Only the uploader can grant access");
8           require(!documents[uuid].hasAccess[grantee],
9           "Access already granted");
10
11          documents[uuid].hasAccess[grantee] = true;
12          documents[uuid].accessList.push(grantee);
13
14          emit AccessGranted(uuid, msg.sender, grantee);
15      }
16
17      function revokeAccess(string calldata uuid, address grantee)
18      external {
19          require(documents[uuid].isActive,
20          "Document is inactive or does not exist");
21          require(documents[uuid].uploader == msg.sender,
22          "Only the uploader can revoke access");
23          require(documents[uuid].hasAccess[grantee],
24          "No access to revoke");
25
26          documents[uuid].hasAccess[grantee] = false;
27
28
29          emit AccessRevoked(uuid, msg.sender, grantee);
30      }
31
32      function getAccessList(string calldata uuid)
33      external view returns (address[] memory) {
34          require(documents[uuid].uploader == msg.sender,
35          "Only the uploader can view access list");
36          return documents[uuid].accessList;
```

```
37      }
38
39      function hasAccess(string calldata uuid, address user)
40      public view returns (bool) {
41          return documents[uuid].isActive && documents[uuid].hasAccess[user
        ];
42
43      }
44
45  }
```

Listing 7.5: Zugriffsverwaltung

7.2. Backend

The backend consists of several components: a middleware responsible for handling the business logic, an off-chain storage for storing data and documents, an additional database for user data, and a simplified keystore for blockchain identities. The following requirements are supported by the backend:

- Unique document identification (FR1)
- Protection against unauthorized access (NF1)

7.2.1. Keystore

To interact with the blockchain, each user requires access to a blockchain account. Blockchain accounts are based on asymmetric encryption, consisting of a public and a private key [Mey22]. In decentralized systems, users are responsible for managing their own accounts and private keys. However, in enterprise applications, blockchain accounts are typically managed through centralized or custodial solutions to ensure compliance, security, and operational efficiency.

For the proof of concept, the blockchain accounts created in the backend are stored locally as encrypted JSON files according to the Web3 Secret Storage specification **(3)** [1]. This standard defines how private keys for blockchain-based applications are securely stored and encrypted. The management of the passphrase for decrypting the accounts is handled by the user to ensure a quick implementation and enhanced user-friendliness for the PoC. To enable all write operations for the blockchain to be fully processed from the backend, a dedicated backend session is created for each user upon login, in which the passphrase is temporarily stored. This allows automated transaction signing during the user's archive access without requiring individual confirmation for each operation that the user performs. This serves as an interim solution for the PoC to simplify its use. In a production environment, the management of blockchain accounts and the signing of transactions should be handled by a secure, enterprise-managed solution.

[1] Web3 Secret Storage. https://ethereum.org/en/developers/docs/data-structures-and-encoding/web3-secret storage/

With user-side management of the passphrase, there is a risk of data loss if employees leave the company or mishandle their keys. To ensure compliance with audit-proof archiving requirements, it is assumed that the organization can ensure access to the cryptographic keys through internal processes if needed.

7.2.2. Middleware

The middleware **(4)** of the proof of concept is responsible for handling the business logic and acts as the central component for processing archiving requests from the frontend. It also acts as an interface to both the blockchain and off-chain storage. Communication between the frontend, backend, and off-chain storage is facilitated via a REST API, while interactions with the blockchain are executed through the JSON-RPC API (rpcnode).
Since authentication using public blockchain addresses (e.g., x08392e32d23..) is impractical for enterprise applications, an additional centralized user management system has been implemented. This ensures that interactions can be reliably traced back to a specific user, supporting the traceability requirement for audit-proof archiving. Users must log in with a password-protected account to access the system. For the proof of concept (PoC), user data is stored in an SQL database **(5)**. When a new user account is created, a corresponding blockchain account is automatically generated in the backend. Registration requires entering an email address, a password, and an additional passphrase used to generate the blockchain account. Only the hashed password is stored in the database to ensure security. Upon successful login, a JSON Web Token (JWT)[2] is issued by the backend. This token grants authorized access to the system's archiving functions. Additionally, with a valid JWT, a time-limited server-side session is created, temporarily storing the user's passkey for blockchain interactions.

After successful authentication, the user is authorized to store documents in the Archive-Storage. To ensure structured archiving according to **(FR1)**, each document is assigned a unique archive object ID. The system uses Google's Universally Unique Identifier[3] (UUID) package, which adheres to the RFC 4122[4] standard for implementing UUIDs in modern systems. The document is categorized and stored in the off-chain storage based on the user-defined folder structure. The ID also serves as a reference for retrieving the archive data object.
When a document is uploaded, the corresponding archive data objects must be properly encrypted before being stored in off-chain storage [SJB19]. Otherwise, data security would depend solely on the policies set by the off-chain storage provider. The German Federal Office for Information Security (BSI) mentions encryption methods in their technical guideline TR-03125 as an additional measure to enhance data confidentiality **(NF1)** [fSidI19]. To protect data from unauthorized access, the proof of concept implements a hybrid encryption scheme. The approach combines the strengths of symmetric and asymmetric encryption and is well-suited for securing data in cloud environments [KK23]. Symmetric encryption relies on a single secret key used for both encryption and decryption.

[2] JWT. https://auth0.com/docs/secure/tokens/json-web-tokens
[3] UUID. https://github.com/google/uuid
[4] RFC 4122. https://datatracker.ietf.org/doc/html/rfc4122

It offers high efficiency and speed, making it especially suitable for handling large amounts of data. However, it has security limitations compared to asymmetric methods, which use a key pair consisting of a public key for encryption and a private key for decryption. These asymmetric methods are considerably slower and more resource-intensive, and are therefore not ideal for encrypting large data objects. By combining the two approaches, it is possible to leverage both the speed of symmetric encryption and the security of asymmetric encryption [KK23]. In the system, documents are first encrypted symmetrically using the AES-256 algorithm as a standard. To protect the AES keys, the hybrid ECIES (Elliptic Curve Integrated Encryption Scheme) method is used. Since both ECIES and blockchain account keys are based on elliptic curve cryptography (ECC), thesame key pairs can be reused to encrypt the AES key. As a result, the encrypted AES key can be securely stored in a database, while decryption of the documents is only possible to users holding the corresponding private key of their blockchain account.
The middleware is accessible as a Docker container via port 9095.

- middleware, port: 9095:9095

7.2.3. Off-Chain Storage

The following requirement is supported with off-chain storage:

- Retention (FR2)

User-Storage For the Proof of Concept (PoC), a PostgreSQL database was implemented to store user-related data **(5)**. Along with credentials, the database also stores encrypted AES keys as well as a reference to each user's blockchain account. The latter enables the assignment of archival operations to an individual user, which is essential for the requirement of traceability. For many enterprises, maintaining control over sensitive user information is essential for ensuring compliance with data protection regulations such as the GDPR. This design choice supports such control while also offering greater flexibility in selecting archival storage solutions.
The User-Storage is accessible as a Docker container via port 5432.

- userDB, port: 5432:5432

Archive-storage The Google Cloud off-chain storage **(6)** is responsible for securely storing archived documents. To ensure confidentiality, all data is encrypted before being stored, preventing direct access to documents from the storage layer itself. The backend communicates with Google Cloud Storage (GCS) through authorized API calls. For the PoC, the cloud storage was configured as single-region storage with the standard storage class, as this option was available free of charge. However, for production use, a multi-region storage with the archive storage class is recommended to provide greater resilience against data loss and improved availability. In particular, backup and recovery

strategies remain an area of future research to fully meet the requirements for data security **(NF6)** and availability **(NF4)**. Utilizing cloud storage, the system can efficiently handle large volumes of data **(FR2)**, while the smart contract implementation in the PoC ensures compliance with retention periods for the users of the system. To prevent unnoticed deletions, even in the event of administrator access to Google Cloud Storage, object retention locks are applied upon upload based on the retention periods defined in the smart contract. The configuration in Google Cloud Storage serves as an additional technical measure to reinforce data retention. As a result, even administrators are unable to delete data directly from the Archive Storage before the retention period expires.

7.3. Frontend

The frontend **(7)** supports the following requirement:

- Availability (NF4)

In the Proof of Concept (PoC), a graphical user interface enables users to retrieve data from the archive system at any time, in accordance with the availability requirement **(NF4)**. Beyond retrieval, the interface also allows users to interact directly with the archive system. The frontend provides seamless access to the platform, allowing users to register, log in, and make use of core archiving functionalities. Users can define filing structures and perform essential operations such as securely storing and retrieving documents, as well as deleting them once the defined retention period has expired. To ensure the integrity of archived data, a "verify" function is integrated into the interface, enabling users to validate stored documents throughout the retention period. Access rights can also be managed directly by granting or revoking permissions to the data objects. Finally, the frontend also offers access to audit logs that record all interactions with the individual archival data.
The frontend is available as a Docker container and can be accessed via port 5173.

- frontend, port: 5173:5173

7.4. Setup

The project is designed so that all system components, including the blockchain network, are automatically initialized.

1. Navigate to the archiveChain directory and execute the command: *docker compose up -d*.
2. An RPC node, four validators, associated monitoring and logging services, as well as traditional components such as the frontend, middleware, and the PostgreSQL database, will start automatically. It is recommended to wait 30 seconds during the initial application startup to allow the network to be fully configured.

3. The backend automatically deploys a smart contract via the JSON-RPC API on the blockchain if it does not already exist.

4. If the smart contract was newly deployed, its address is stored in the .env file, requiring the middleware to be restarted in order to access it. Execute: *docker restart archivechain middleware-1*.

5. The system can now be accessed via the frontend in a browser at localhost:5173. Alternatively, interactions with the backend API can be performed using Postman.

8. Use Case

This section outlines three specific use cases in which the implemented system is applied: uploading, retrieving, and deleting documents from the archive system. These use cases illustrate the practical workflows and the benefits of the system in a real-world enterprise context. They also demonstrate that the functionalities described in Chapter 5.2 have been successfully implemented within the system.

Szenario

A medium-sized company in the manufacturing industry is required to retain its financial documents in accordance with legal regulations for audit-proof archiving. Financial documents such as invoices, receipts, balance sheets, and tax returns must be securely archived, easily retrievable, and properly deletable. During the retention period, these documents must not be altered or deleted, and access must be restricted to authorized individuals only. To ensure traceability and verifiability, a complete documentation of interactions with the archived data objects must be maintained.

Given that the use of legally required retention periods would be impractical for the presented use cases, the retention period was reduced to a maximum of twenty minutes, depending on the document category.

Preconditions

The company is authorized by the blockchain operator to participate in the network, and a smart contract has been deployed on the blockchain as described in Section 7.4. To interact with the system, an employee logs in via the browser, as shown in Figure 8.1. If no account exists for the archive system, the user can create one. Upon entering an email address and password, a JWT token is generated by the backend, which authorizes the user for subsequent requests within the system.

During initial registration, the user is asked to enter a passphrase, which is used to generate a user-specific blockchain identity in the backend. Upon successful creation, the user is redirected to the dashboard page of the frontend, as shown in Figure 8.2.

8.1. File Upload

The upload of financial documents represents the first use case within the archiving process. Initially, the employee creates a filing structure by adding a folder for the document to be archived. In this example, the folder is named Invoices. After selecting the folder, the

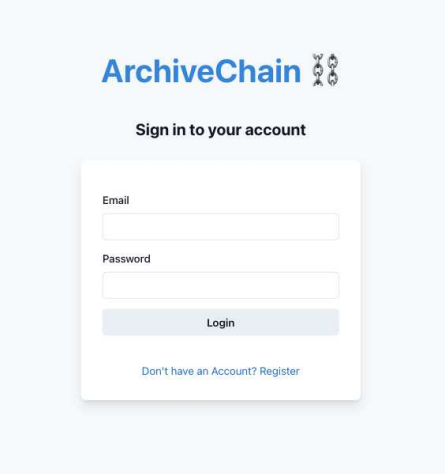

Figure 8.1.: ArchiveChain Login

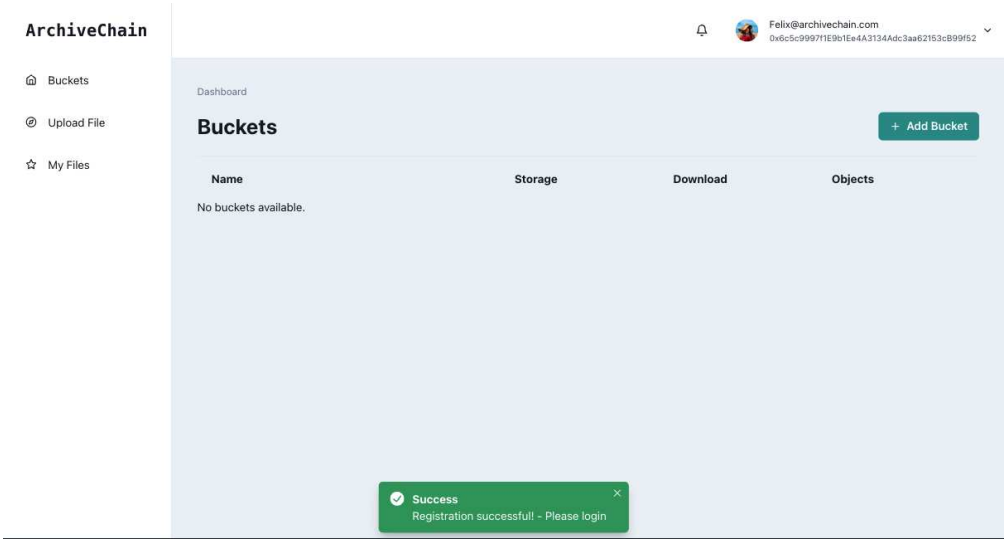

Figure 8.2.: ArchiveChain Dashboard

user can initiate the upload process to archive a document. If the upload button is not displayed immediately, the option Upload File can be accessed via the sidebar.

Use Case:	**File Upload**

Goal: Storage of a document in the archiving system.

Primary Actor:

- **User:** Interacts with the system to upload a document.

System Components:

- **Frontend:** Provides the user interface for initiating the upload process.
- **Backend:** Handles the upload logic, generates file hashes, extracts metadata, and encrypts documents before they are stored.
- **Keystore:** Stores the user's encrypted blockchain keys, used for transaction signing.
- **Smart Contract:** Provides functions for verifying document integrity, storing metadata as well as access rights and retention periods.
- **Blockchain:** Stores the document related data immutably.
- **User-Storage:** Stores user credentials and encryption keys.
- **Archive-Storage:** Stores the encrypted document and applies retention policies.

Description: An employee of the organization wants to upload a document to the system and archive it in an audit-proof manner.

Preconditions:

- The DocumentStorage smart contract is deployed on the blockchain.
- The user is registered and authenticated in the system.
- The user has created a filing structure for the document via the frontend.

Steps:

1. The user selects a document for upload and specifies its category.
2. The frontend checks the document and generates a SHA-256 hash of the file before uploading.
3. The backend extracts relevant metadata from the document. A retention period is assigned based on the document category.
4. The backend generates a SHA-256 hash of the received document and its metadata.
5. A unique identifier (UUID) is created for the document.
6. The generated hash values and the defined retention period are transmitted to the smart contract (/uploadDocument).
7. The smart contract verifies the consistency of the hashes generated in the frontend and backend to ensure the integrity of the document before storage (/verifyDocument).
8. The smart contract assigns the uploader as the authorized user for access to the document.
9. The hash values of the document and metadata, along with the retention period, are immutably stored on the blockchain.
10. The upload interaction is traceably and immutably recorded in the blockchain's transaction logs via the *DocumentUploaded* event.
11. The backend encrypts the document and its metadata using an AES key to securely store it in the Archive-Storage.
12. The AES key is encrypted using the public key of the user's blockchain account via ECIES and then stored in the User-Storage.
13. The backend transmits the encrypted document and metadata, along with the defined retention period, to the Archive-Storage.

14. The Archive-Storage applies the defined object retention lock and stores the document in the cloud storage.

Postconditions:

- The user receives a confirmation of the successful upload, including the blockchain transaction details.
- The document is securely stored in the Archive-Storage and can no longer be deleted or modified without detection.
- The document hash and metadata are immutably and securely recorded on the blockchain.

Exceptions:

- The smart contract detects a hash conflict and notifies the user that the file may have been manipulated.
- The upload to the Archive-Storage fails, and the user is informed via an error message.

Requirements:

- **FR1:** Each document is uniquely identifiable and stored in a structured manner.
- **FR2:** Documents are securely and immutably stored for a defined retention period.
- **FR4:** The interactions are traceably documented on the blockchain.
- **NF1:** Data is protected through smart contract-based access control and hybrid encryption.
- **NF2:** Access rights to archived documents are defined during upload through the smart contract.
- **NF3:** Data integrity is ensured by comparing the hash value before the upload with the hash value stored on the blockchain for the archived document.
- **NF5:** Immutability is guaranteed through the unalterable storage of document hashes and metadata on the blockchain.

If all processes are completed successfully, the user receives a confirmation in the frontend, including the transaction details on the blockchain. Figure 8.3 shows the successful upload of the document `01_Notion Invoice.pdf` in the archive.

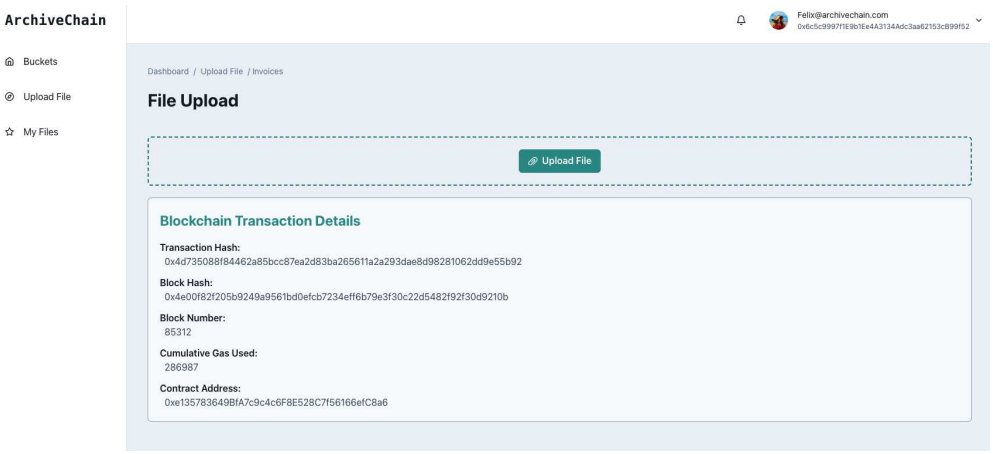

Figure 8.3.: ArchiveChain Successful File Upload

This can also be verified using the service *Quorum Explorer* [1]. It provides an overview of recent blocks and transactions. Using the transaction hash, the corresponding transaction can be traced on the blockchain. Figure 8.4 shows the transactions related to the two smart contract functions `/verifyDocument` and `/uploadDocument` that were executed during the upload process.

To ensure that the document was correctly stored in the Archive-Storage, the archive is accessed via the Google Cloud Platform. Figure 8.5 shows that the document was saved within the user-defined filing structure and that the retention period was properly set. In this example, the retention period ends on July 1, 2024, at 00:16. The defined retention time in this case is one minute, during which the record's preservation in the Archive-Storage is guaranteed.

Due to encryption, the document cannot be accessed without authorization, not even by the administrator of the configured Google Cloud Storage. Figure 8.6 shows an attempt to access the file as an administrator via the Google Cloud Platform, but the document is not readable for the user.

By switching to the "My Files" tab, the user is able to view the archived document. The green verified checkmark indicates that the document has been successfully validated

[1] Quorum Explorer. https://github.com/Consensys/quorum-explorer

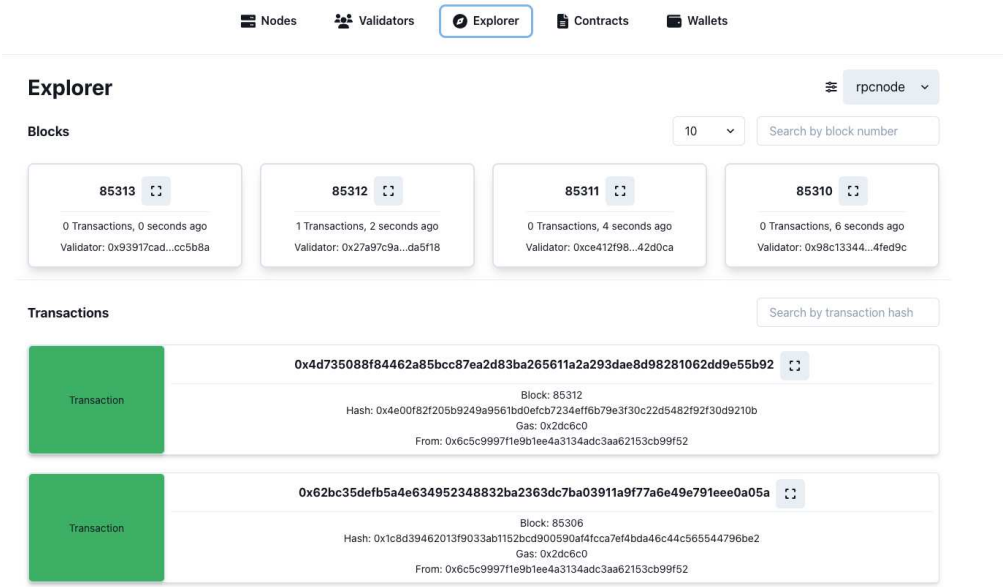

Figure 8.4.: Quorum Block Explorer Confirmation

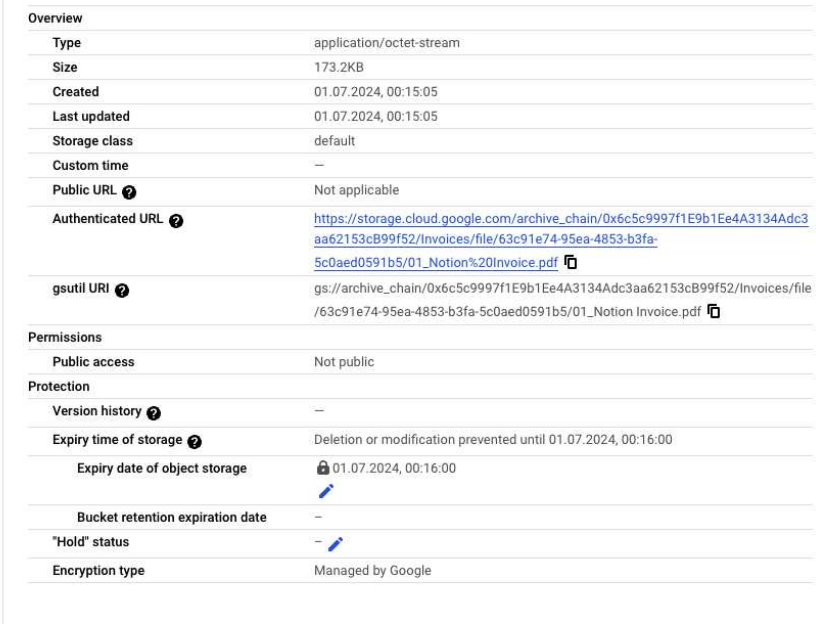

Figure 8.5.: Google Cloud Storage Confirmation

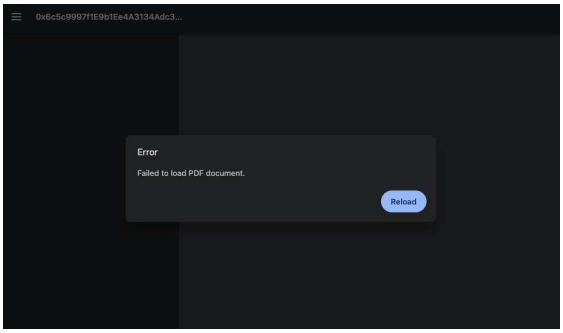

Figure 8.6.: Encrypted Download

on the blockchain, confirming that the archived document matched the original at the time of submission. By clicking the three dots at the right end of the document box, users can perform an additional integrity check during the retention period using the "Verify" function. This checks whether the document has remained unchanged by comparing the current file hash in the Archive-Storage with the reference value stored on the blockchain. It provides proof that the data has been stored correctly and unaltered up to that point. The "View Logs" function allows users to retrieve all events emitted by the smart contract (such as "FileVerified" and "FileUploaded") along with their associated transaction details, supporting traceability. Figure 8.7 shows the transactions resulting from the file upload.

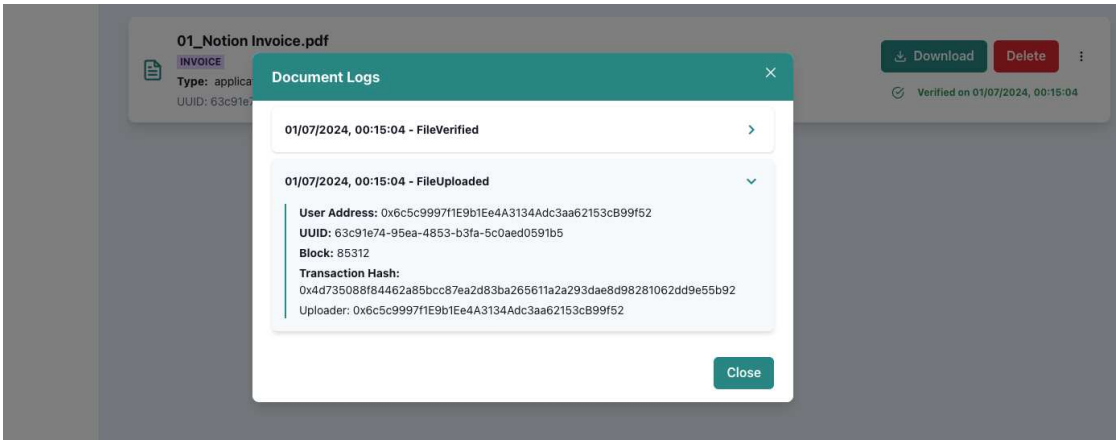

Figure 8.7.: ArchiveChain File Logs

In summary, the file upload use case demonstrates the secure process of uploading and storing financial documents within the designed archiving system. The process ensures that documents are stored securely and without modification, while all interactions are transparently recorded on the blockchain.

8.2. File Retrieval

The retrieval of archived financial documents is essential for meeting the requirement of availability, which demands that data remain continuously readable and retrievable. This process must also ensure that only authorized users are granted access to the archived documents. To illustrate the functionality of the access control mechanism in more detail, the authorization process is explained in Subsection 8.2.1.

Clicking the "Download" button initiates the process via the user interface.

Use Case: **File Retrieval**

Goal: Authorized retrieval and readable presentation of a document from the Achive-Storage.

Primary Actor:

- **User:** Initiates the document download via the frontend.

System Components:

- **Frontend:** Provides the user interface for initiating the download request.
- **Backend:** Processes the download request, coordinates the interaction with other system components, and decrypts the document.
- **Keystore:** Stores the user's encrypted blockchain keys, used for transaction signing.
- **Smart Contract:** Verifies access rights and ensures the integrity of the requested document.
- **Blockchain:** Stores the immutable reference data of the document.
- **User-Storage:** Provides the user's encrypted AES-key required for decryption.
- **Archive-Storage:** Provides the encrypted document.

Description: An employee of the organization wants to retrieve a document from the system and access it in a readable format.

Preconditions:

- The user is registered and authenticated in the system.
- The smart contract is deployed on the blockchain.
- The document to be retrieved has already been archived in the system.

Steps:

1. The user initiates the download request via the frontend.
2. The backend retrieves the encrypted document and its metadata from the Archive-Storage.
3. The associated AES key is fetched from the User-Storage and decrypted using the

user's private blockchain key via ECIES.

4. The backend decrypts the document and metadata using the decrypted AES key and generates a file hash.
5. The generated hash and the document UUID are submitted to the smart contract.
6. The smart contract verifies whether the user is authorized to access the document (/hasAccess).
7. The smart contract checks the integrity of the document by comparing the hash against the reference value stored on the blockchain (/verifyDocument).
8. The retrieval interaction is traceably and immutably recorded in the blockchain's transaction logs via the *DocumentAccessed* event.
9. The decrypted document is made available for retrieval.

Postconditions:

- The user receives a success message in the frontend indicating that the download has started.
- The document has been successfully retrieved by the user.

Exceptions:

- Access is denied if the user does not have the required permissions.
- The smart contract detects a hash conflict and informs the user that the file may have been altered during the retention period.
- Retrieval from Archive-Storage fails, and the user is notified via an appropriate error message.

Requirements:

- **FR4:** The access interaction is transparently recorded on the blockchain.
- **NF2:** The retrieval of archived data is limited to users with appropriate access rights
- **NF3:** The integrity of the data is ensured by comparing the file hash in the blockchain with that of the archived object.
- **NF4:** The document is made readable through the retrieval process.

Upon successful completion of the process, the file is made available for retrieval. The corresponding interaction is transparently recorded and can be reviewed through the "View Logs" function.

8.2.1. Share Access

By default, the smart contract automatically grants access rights to the uploading user. This assignment can be verified via the "Check Access" function in the frontend, which displays the public address of the user's blockchain account. If an employee wishes to grant access to a document to another colleague, the process can be initiated using the "Grant Access" function. The only requirement is the email address of another user who is already registered in the system. It is important to note that access can only be granted if the recipient has previously interacted with the blockchain, for example by uploading a document, so that their public blockchain address is known to the system. Once the email address is entered, the process is initiated on the backend. The backend identifies the public key of the recipient's blockchain identity based on the provided email. Using this public key and the document's UUID, the smart contract is called to register the recipient as an authorized user for the document. In order for the recipient to get access to the actual file, the AES key used for encryption must also be securely shared. For this purpose, the uploader's AES key is first decrypted and then re-encrypted using the recipient's public key via ECIES. The newly encrypted AES key is then stored in the User-Storage for the recipient.
Figure 8.8 shows the confirmation displayed in the frontend, which is also transparently verifiable via the event logs. In this example, access was granted to the email address *Test@archivechain.com*. After access is granted, the recipient's public blockchain address is displayed in the list of authorized users under the "Check Access" function in the frontend.

The function for retrieving data by a newly authorized user was not implemented in the frontend. Instead, access is possible via the API using Postman. To do this, the following environment variables must be adjusted in the provided Postman collection:

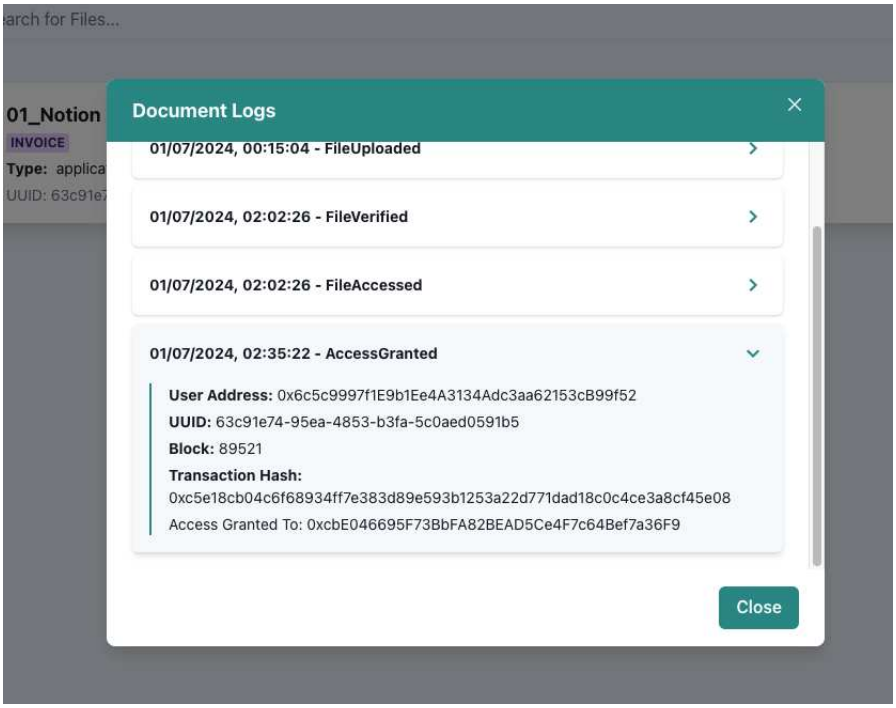

Figure 8.8.: ArchiveChain Access Granted Logs

- `password`
- `passphrase`
- `email`
- `filename`
- `subFolderName` (The name of the folder where the file was initially stored)
- `uuid` (The UUID of the requested file)
- `publicAddressSharedUser` (The public address of the uploader)

The following functions must then be executed in the order shown below to retrieve the file for the newly authorized user:

1. `LoginUser`
2. `CreateSession`
3. `DownloadFile`

Figure 8.9 shows the confirmation of a successful file retrieval by the recipient *Test@archivechain.com* for the file originally uploaded by *Felix@archivechain.com*.

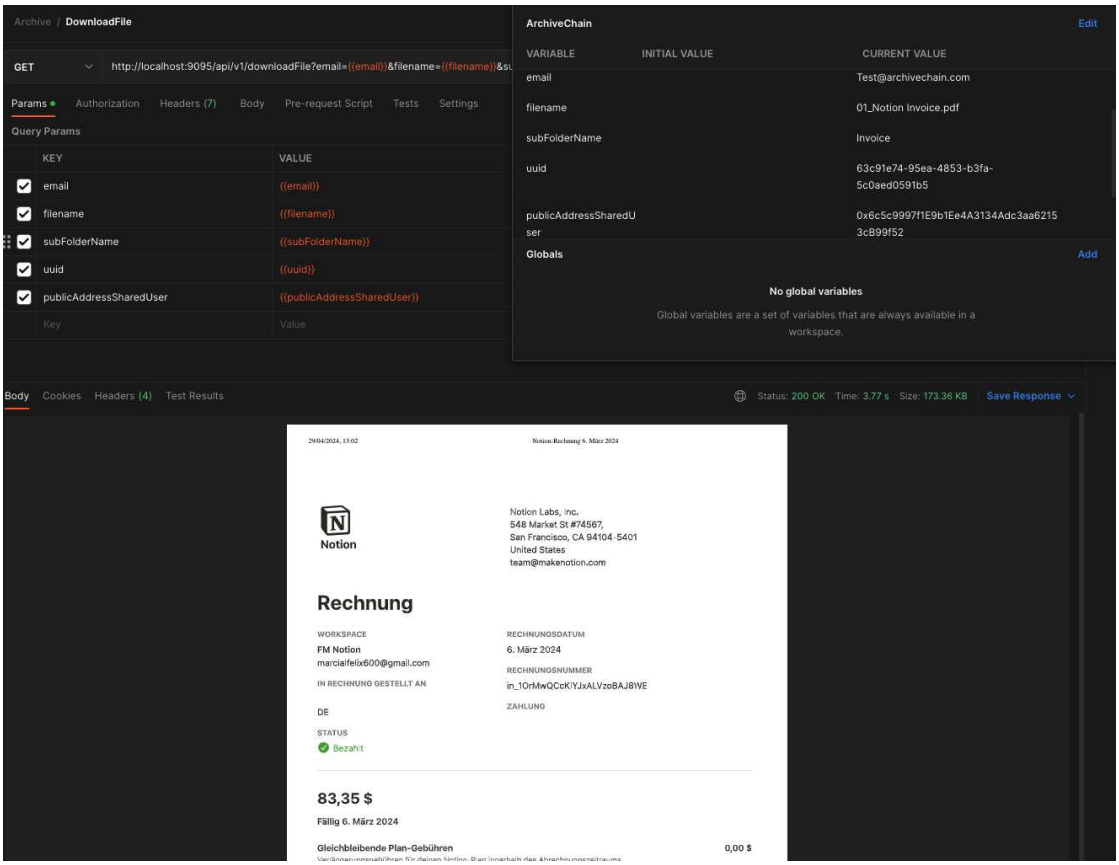

Figure 8.9.: ArchiveChain: Retrieving a shared file

To revoke access, the uploader can use the "Check Access" function in the frontend and click the red X icon next to the recipient. Any subsequent access attempt will fail, as the transaction will be reverted by the smart contract due to missing permissions.

```
{
    "code": 500,
    "message": "Execution reverted: transaction failed with receipt status 0",
    "status": 500,
    "error": "Internal server errors"
}
```

Listing 8.1: No Access

The file retrieval use case demonstrated that files are only made retrievable and readable to users who have been granted access rights to the archived document. Document integrity and availability were verified, and all interactions with the archived data object were logged in support of the requirements for traceability and verifiability.

8.3. File Deletion

The compliant deletion of financial documents requires that files can only be deleted after the expiration of the legally defined retention periods from the archive.

Use Case:	**File Deletion**
Goal:	Proper deletion of a document after the end of the retention period.
Primary Actor:	• **User:** Initiates the deletion of a document via the user interface.

System Components:

- **Frontend:** Provides the interface for selecting and initiating the deletion of a document.
- **Backend:** Processes the deletion request, interacts with the smart contract to verify retention expiry, and initiates the document deletion from the Archive-Storage.
- **Keystore:** Provides the user's private key for transaction signing
- **Smart Contract:** Checks access rights and retention periods, and marks the document as deleted on-chain.
- **Blockchain:** Immutably stores data related to the deletion process
- **Archive-Storage:** Deletes the off-chain document once the defined conditions have been successfully verified.

Description: An employee of the organization wants to delete a document from the system after the retention period has expired.

Preconditions:

- The user is registered and authenticated in the system.
- The smart contract has been deployed on the blockchain.
- The user previously uploaded the document with a defined retention period.
- The document is stored in the Archive-Storage, and a hash reference as well as access rights are registered in the smart contract.

Steps:

1. The user initiates a deletion request through the frontend using the document's

UUID.

2. The backend forwards the request to the smart contract.
3. The smart contract verifies whether the retention period of the given document has expired and checks if the user has the required permissions.
4. If the conditions are met, the document is marked as inactive in the smart contract.
5. The deletion interaction is traceably and immutably recorded in the blockchain's transaction logs via the *DocumentDeleted* event.
6. The backend initiates the deletion of the encrypted document and its metadata from the Archive-Storage.
7. The data is deleted from the Archive-Storage.

Postconditions:

- The user receives a success message in the frontend confirming the deletion.
- The document and metadata have been properly deleted after the end of the retention period from the Archive-Storage.

Exceptions:

- If the smart contract determines that the retention period has not yet expired, the deletion request is rejected.
- If the user lacks the necessary permissions, the deletion process is not executed.
- If deletion from the Archive-Storage fails, the user is informed with an appropriate error message.

Requirements:

- **FR2:** Documents can be properly deleted after the expiration of their defined retention periods.
- **FR4:** The interactions are traceably documented on the blockchain.
- **NF2:** Only users with access rights to a document are authorized to initiate its deletion

Upon successful deletion, the financial document `01_Notion Invoice.pdf` was marked as inactive on the blockchain and removed from the Archive-Storage. While the retention period is already enforced at the smart contract level, the defined object retention lock in the Archive- Storage prevents premature deletion attempts even by administrators. Figure 8.10 illustrates the error message displayed when a deletion attempt is made prematurely. In this example, the retention period is set to *2024-06-30T15:16:00-07:00*, which corresponds to July 1, 2024, at 00:16 (CEST, Vienna time). The deletion attempt at 00:15 was therefore unsuccessful.

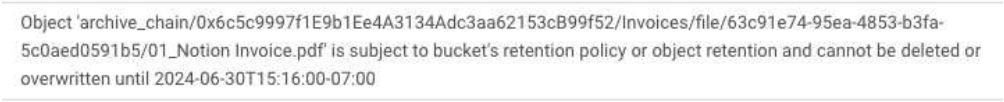

Figure 8.10.: Google Cloud Platform: Deleting before retention period is met

Since the deletion process was also recorded on the blockchain, it can be traced by entering the file's UUID in the top-right corner of the frontend interface (see Figure 8.11).

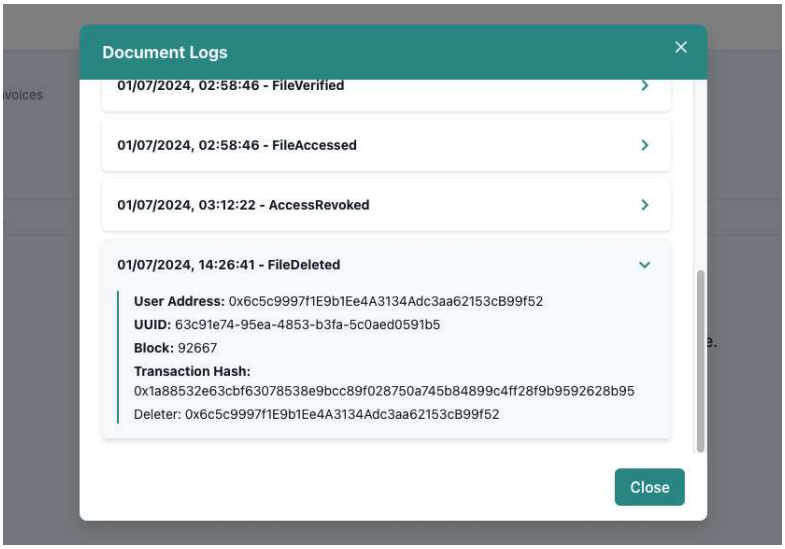

Figure 8.11.: ArchiveChain: Document deleted logs

Overall, the file deletion use case illustrates the proper removal of archived financial documents. Deletion is strictly bound to the expiration of the retention period as well as the access rights defined in the smart contract, while all interactions with the data object are immutably logged to support the requirements for traceability and verifiability.

9. Evaluation

This chapter critically evaluates the prototype implementation with regard to the requirements of an audit-proof archiving system. Specifically, it examines how the developed system contributes to answering the research questions defined in Chapter 3. The evaluation focuses on the extent to which the defined requirements were fulfilled through the use of blockchain and smart contracts, highlighting specific benefits, limitations, and open challenges of the implementation. In addition, a short evaluation of the user experience (UX) is presented to capture the usability of the prototype from a user's perspective, followed by an analysis of the usage costs associated with operating the system.

9.1. Fulfillment of Requirements

With the given implementation, it was shown that blockchain technology can effectively support essential requirements of audit-proof archiving by serving as an integrity layer for the underlying system. While not all requirements can be fulfilled solely through blockchain, the prototype demonstrates that essential aspects of compliance can be addressed within a hybrid system architecture. The following table 9.1 provides an overview of the legal requirements considered in this thesis and shows the extent to which they are supported by the use of blockchain technology in the implemented system:

Table 9.1.: Overview of Supported Requirements

Requirement	**Supported**
FR2: Retention	(✓)
FR4: Traceability & Verifiability	(✓)
NF1: Confidentiality	(✓)
NF2: Authorization	✓
NF3: Integrity	✓
NF4: Availability	✗
NF5: Immutability	✓

✓ indicates that the requirement is supported. (✓) indicates that the requirement is partially supported. ✗ indicates that the requirement is not supported.

Several key requirements are supported either fully or partially by blockchain technology

in the prototype, particularly those concerning integrity, immutability, and authorization. However, other areas reveal the limitations of the blockchain-based approach. The requirements that were either fully or partially supported were addressed through four core blockchain functions integrated into the prototype. Building on the conceptual design outlined in Section 5.2, each of these functions was developed to address one of the sub-questions **RQ1.1** to **RQ1.4**. Table 9.2 maps these functions to the corresponding research questions.

Table 9.2.: Assignment of Functions to RQs

Function	**RQ**
Storing cryptographic hash values on-chain	RQ1.1
Verifying and managing access rights to archival data	RQ1.2
Enforcement of retention periods	RQ1.3
Immutable logging of document interactions	RQ1.4

In the following sections, each research question is evaluated based on the implemented function it relates to. The evaluation considers which compliance requirements are addressed, how they are supported through blockchain technology, and to what extent this contributes to fulfilling the legal objectives of audit-proof archiving.

RQ1.1 – How can blockchain be used to ensure the integrity of archived data?

The requirement of integrity (**NF3**) refers to the ability to verify that all archived information remains complete, accurate, and free from unauthorized modification. The developed prototype supports this requirement by immutably storing cryptographic hash values of each archived file and its metadata on the blockchain. These hashes act as digital fingerprints that can be used to verify whether a document has remained unchanged throughout its retention period. In the prototype implementation, upon uploading a document, an SHA-256 hash is generated to uniquely represent the content of the file and its metadata. This hash is stored on-chain via a smart contract and serves as a tamper-evident reference value. However, the reliability of this verification mechanism depends on the integrity of the reference data itself. If the stored hash could be modified after the fact, the system would no longer be able to prove whether a document was changed. This is where the immutability of the blockchain becomes essential. The blockchain's immutability both as a technical property and as a compliance requirement (**NF5**) ensures that archived data cannot be suppressed, overwritten, or falsified without detection. Once a hash is stored on-chain, it can't be retroactively modified. As a

result, the blockchain functions as an immutable source of truth. To perform integrity verification, the system compares the current hash of the document retrieved from the Archive-Storage with the original reference value on the blockchain. Any deviation between the two indicates a modification and, consequently, a loss of integrity. This comparison is executed automatically upon access and can also be triggered manually through a dedicated verification function provided by the smart contract. While the blockchain guarantees the immutability of the reference data, the actual document is stored outside the blockchain. As a result, the system cannot prevent manipulation of the stored file, but it can reliably detect it. This design allows for verifiable proof of integrity, even in cases where the archive storage is compromised.

Overall, blockchain technology can support data integrity by immutably storing cryptographic hash values of archived documents and metadata on-chain. This ensures that the reference values remain unchanged over time, providing a reliable basis for verifying the integrity of archived data throughout the retention period. The compliance requirements **NF3** and **NF5** can be considered fulfilled through the use of blockchain technology.

RQ1.2 - How can blockchain technology be used to support the requirements related to secure access of archived information?

Secure access to archived data requires that only authorized users can retrieve or process documents (**NF2**) and that the data is protected against unauthorized viewing or disclosure (**NF1**). These two requirements are addressed in the prototype through the use of smart contracts and a permissioned blockchain infrastructure. The requirement of authorization (**NF2**) is fulfilled by managing access permissions directly on the blockchain. For each document, an access list is maintained in the smart contract, defining which users are allowed to retrieve or verify the corresponding file. Permissions can be granted or revoked by the document owner through dedicated functions, and each action is recorded immutably on the blockchain. The smart contract handles the verification and assignment of access permissions, making access control transparent and tamper-proof. Compared to traditional, centralized access control mechanisms, which are vulnerable to manipulation, the blockchain-based approach increases transparency and reduces the risk of unauthorized changes. However, using smart contracts for access control also introduces certain risks. Since smart contracts are immutable once deployed, any errors in the contract code cannot be corrected afterwards, which may lead to unintended behavior or security vulnerabilities. The requirement of confidentiality (**NF1**) is partially supported. On the one hand, access to documents is restricted to authorized users as defined in the smart contract, which inherently limits exposure. On the other hand, the use of a private, permissioned blockchain (GoQuorum) ensures that only verified participants can interact with the network, providing a base level of infrastructure-level confidentiality. However, since the archived documents themselves are stored off-chain, additional protection is required within the Archive-Storage. This is implemented in the prototype through hybrid encryption, which ensures that even with administrative access to the storage system, document contents remain protected. As described in Chapter 8.1, the encryption mechanism is effective but not blockchain-specific. Since encryption is applied outside

the blockchain, this aspect does not represent added value through the technology itself and corresponds to conventional archiving practices.

To answer **RQ1.2**, blockchain technology supports secure access to archived data by enabling transparent and verifiable permission management through smart contracts. The requirement of authorization (**NF2**) is fulfilled, since access rights are enforced directly on-chain and cannot be manipulated externally. The requirement of confidentiality (**NF1**) is only partially supported. While access to archived data is restricted through the use of a private permissioned blockchain network and smart contract validation, the protection of the actual document content is achieved through conventional encryption outside the blockchain.

RQ1.3 - How can blockchain technology be used to support the requirement for data retention?

Due to the limitations of blockchain in storing large volumes of data, the retention requirement can only be supported in terms of enforcing retention periods. In the implementation, this is achieved through a policy embedded in the smart contract, where the on-chain metadata, including the retention period, is automatically checked before a deletion process is triggered. A deletion is only authorized once the defined period has been exceeded. This decouples the deletion process from manual or organizational approvals and reduces the risk of human error. As a result, premature or unauthorized deletion of documents at the application level is prevented, strengthening the system's overall integrity and reliability. However, the decision to adopt a hybrid architecture, in which archived documents are stored off-chain, introduces a trade-off. While this enables the system to store large volumes of data as required, blockchain does not have direct control over the retention enforcement within the Archive-Storage. Although compliance with retention periods can be technically guaranteed on the user side, the system remains vulnerable to unauthorized deletions performed directly in the Archive-Storage. In the prototype implementation, this was addressed by additionally applying the retention periods defined in the Smart Contract as object retention lock parameters within the Archive-Storage. However, this relies on trusting the storage provider as a third party, and the usefulness of duplicating the enforcement logic both on- and off-chain may be questioned.

In summary, blockchain technology can support the requirement for data retention by reliably enforcing retention periods at the application level and by transparently documenting deletion eligibility. However, since the actual storage and deletion of archived documents take place outside the blockchain, the requirement of retention (**FR2**) can only be partially supported, and the added value of blockchain remains limited.

RQ1.4 - How can blockchain technology be used to support the requirement for traceability and verifiability?

The requirement of traceability and verifiability (**FR4**) refers to the ability to track and reconstruct all relevant actions and events within an archiving system in a reliable and

tamper-proof manner. In the developed prototype, blockchain technology enables such traceability by recording key interactions with archived documents directly on-chain. Each relevant action, such as uploading, retrieving, deleting a document, or modifying access permissions, is executed through a smart contract and automatically logged as a blockchain transaction. Each transaction includes a timestamp, the triggering user's public address, and the invoked function. Since all transactions are digitally signed and immutably stored on the blockchain, they cannot be changed or deleted afterwards. This creates a verifiable audit trail that documents who did what and when, providing strong evidence of system behavior over time. Compared to traditional logs that are stored in databases, this approach offers a higher degree of integrity. However, the logging functionality is inherently limited to interactions that occur through the smart contract. Actions that take place in other parts of the system, such as frontend operations, backend logic, or activities within the off-chain Archive-Storage, are not recorded on-chain and are therefore not covered by the blockchain's immutability guarantees. As a result, the traceability is only as complete as the scope of the blockchain integration. To improve traceability beyond the current scope, future system versions could anchor logs from conventional components by storing cryptographic hash values of external log files on the blockchain. This would allow for the verification of logs without storing all content on-chain. Additionally, full compliance with the requirement of traceability and verifiability depends not only on technical logging but also on comprehensive procedural documentation. This includes describing how each process is executed, monitored, and maintained across the entire system.

With regard to **RQ1.4**, the system demonstrates that blockchain technology can support the equirement for traceability and verifiability (**FR4**) by providing immutable, time-stamped, and user-specific logs of key actions. This offers a clear advantage over traditional, mutable system logs. However, the requirement itself is only partially fulfilled, as currently, only blockchain-managed operations are recorded. In addition, fulfilling the requirement requires a comprehensive procedural documentation describing how processes are executed and monitored across the entire system.

9.2. User experience evaluation

This section evaluates the user experience (UX) of the developed prototype. The goal is to assess how the system is perceived by users. Since the system is intended for use in a compliance-relevant context, it is important to identify potential negative user experiences early to ensure user acceptance. The usability of the system was evaluated using Brooke's System Usability Scale (SUS), which is one of the most widely used questionnaires in usability studies [Bar20, B+96].

9.2.1. Methodology

The System Usability Scale (SUS) is a simple method for measuring the perceived usability of systems. It was developed by John Brooke in 1986 and has since become an

internationally recognized instrument. Due to the fact that the test is relatively short and produces reliable results even with a small number of participants, it has become particularly popular [TS+04].

It consists of ten questions, with five statements worded positively and the other five negatively. The responses are collected on five-point Likert scales ranging from 1 (Strongly Disagree) to 5 (Strongly Agree), see Figure 9.1 [TA13].

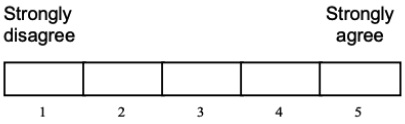

Figure 9.1.: 5 point Likert scale [B+96]

In order to calculate the SUS score, the responses to each question are first converted to values between 0 and 4. For the positively worded items (1, 3, 5, 7, 9), the contribution is the response value minus 1. For the negatively worded items (2, 4, 6, 8, 10), it is 5 minus the response value. The contributions are then summed and multiplied by 2.5, resulting in an overall score between 0 and 100 [B+96]. A SUS score of 70 is generally considered the threshold between good and below-average usability [TA13, BKM08].

Brooke proposed the following ten questions to measure the subjective perceived usability:

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

Based on these questions, the prototype's user interface was evaluated with five participants. They were asked to complete the central use cases of the system, which include uploading a file, file retrieval, and file deletion. Following the completion of the tasks, the SUS questionnaire was conducted in line with Brooke's recommendation. Participants were instructed to record their immediate response to each item rather than overthinking their answers. If they felt unable to respond to a question, they were advised to choose "3" on the Likert scale for that specific question [Bar20, B+96].

9.2.2. Results

The results of the questionnaire made it possible to obtain a quantitative impression of the system's usability. In total, five participants completed the questionnaire. In selecting participants, both technical and non-technical users were included to reflect the expected range of future users of the system. Three of them had an IT background, including one frontend developer and one expert in legally compliant archiving systems, while the remaining two participants had no IT background.

Overall, the prototype received a usability score of 75, which indicates good usability. Since it is still only a prototype, this result is promising but also shows room for improvement. When comparing user groups, participants with a technical background reported higher SUS scores (72.5–97.5), suggesting that they experienced the system as more usable. Within this group, the frontend developer provided slightly lower scores compared to the other technical participants, which may reflect higher expectations regarding user interface and interaction design. In contrast, participants with a non-technical background reported lower SUS scores (57.5 and 70.0), indicating that this group faced more difficulties in using the application.

Table 9.3 presents the average score of the individual questions. Looking at these results, it can be seen that the participants generally agreed that the system was easy to use (4.2) and that its functions were well integrated (4.6). At the same time, lower ratings on confidence while using the system (3.6) and a relatively high perceived need for technical support (2.6) indicate that some users, particularly those with less technical expertise, still faced uncertainties. The scores also show that the system was not considered unnecessarily complex (1.6) or inconsistent (1.6), which suggests that the system provides a clear and well-organized structure.

To sum up, the results show that the application already offers good usability, especially for technical users, as some elements may be more intuitive for them. However, the findings also point to potential barriers for non-technical users, suggesting that additional onboarding support and clearer instructions could improve accessibility and overall usability for a broader audience. To understand the specific challenges faced by this group, further qualitative studies are needed.

Table 9.3.: Average SUS scores per question across all participants

I think that I would like to use this system frequently.	3.8
I found the system unnecessarily complex.	1.6
I thought the system was easy to use.	4.2
I think that I would need the support of a technical person to be able to use this system.	2.6
I found the various functions in this system were well integrated.	4.6
I thought there was too much inconsistency in this system.	1.6
I would imagine that most people would learn to use this system very quickly.	4.2
I found the system very cumbersome to use.	1.8
I felt very confident using the system.	3.6
I needed to learn a lot of things before I could get going with this system.	2.2

9.3. Usage Costs

Using the blockchain as an integrity layer causes additional costs for an audit-proof archiving system. In this section, only blockchain-specific resources are considered, while components such as storage, frontend, or middleware are not included. As the implementation is a private zero-gas network, there are no direct monetary fees per transaction. Instead, costs occur during ongoing operation in the form of additional computation time, network traffic, and the growing storage requirements for blockchain data.

To determine these costs, two measurement windows of ten minutes each were considered: one idle window without transactions and an active window in which a total of 39 transactions were carried out. The results are shown in Table 9.4.

Table 9.4.: Measurements in idle and active phases and resource share per transaction

	Idle	Active	Difference	Per Tx
CPU time [s]	45.971107	51.557522	5.586415	0.143241
Network [B]	2,741,327	4,775,280	2,033,953	52,152.641
Chain storage [B]	1,995,180	2,138,908	143,728	3,685.333

Considering only idle operation, the chain grows by about 8.6 GB per month in this configuration. Each transaction then adds an average of 0.143 s of CPU time, about 52 kB of outgoing network traffic, and roughly 3.7 kB of additional chain data. To illustrate the

cumulative effect under higher load, a total of one million transactions would correspond to approximately 39.8 h of CPU time, 52.15 GB of network traffic, and 3.7 GB of extra storage on top of the baseline. These figures describe the technical resource demand. To translate them into realistic costs, an exemplary deployment environment is required. For illustration, a cloud setup with four validator nodes and one RPC node on *e2-standard-2* instances would incur monthly fixed costs of about 280 Euro in Google Cloud[1]. In such a setup, the main costs come from the infrastructure that must run continuously, while CPU and network usage are already included in the instance price. Variable costs arise primarily from the ongoing growth of blockchain data, since additional storage must be kept permanently. In Google Cloud, this storage is comparatively inexpensive at about 0.041 Euro per GB per month[2], meaning that the additional storage demand leads to steadily increasing but overall manageable usage costs, even in the long run.

9.4. Discussion

This chapter reflects on the evaluation findings and offers a critical interpretation of the conceptual and technical implications of the developed prototype. The discussion aims to assess the practical relevance of blockchain integration beyond the fulfillment of individual requirements and to identify the architectural trade-offs, limitations, and potential of the proposed approach in the context of audit-proof archiving.

It was shown that blockchain technology can provide meaningful support for selected compliance requirements in the context of audit-proof archiving. The most immediate benefit lies in enhancing the integrity of archived data by immutably storing cryptographic hash values and metadata on-chain, which enables the reliable detection of any subsequent manipulation in the off-chain storage. In addition, smart contracts enforce defined retention periods before deletions are permitted, contributing to compliance with legal storage obligations. Access rights are managed transparently and verifiably on the blockchain, ensuring that only authorized users can access archived content and thereby supporting confidentiality and authorization. Finally, the tamper-proof logging of user actions directly on-chain strengthens traceability and verifiability, offering advantages over conventional, mutable logging systems.

Despite all these benefits, the evaluation highlights that blockchain cannot function as a standalone archival solution. Due to limitations in scalability and storage capacity, a hybrid system architecture remains necessary. While the blockchain component ensures integrity and transparency, actual document storage must be handled off-chain using traditional storage infrastructure. This separation of responsibilities introduces a critical dependency, as the effectiveness of the blockchain-based layer ultimately depends on the reliability and trustworthiness of the off-chain environment. This dependency becomes especially relevant when considering operations that occur outside the scope of the smart contract. Actions such as file deletion, upload failures, or administrative access at the

[1] Google Compute Engine Pricing: https://cloud.google.com/products/calculator

[2] Google Persistent Disk and Hyperdisk Pricing: https://cloud.google.com/compute/disks-image-pricing

storage layer are invisible to the blockchain and can result in inconsistencies between on-chain and off-chain states. As such, the system cannot guarantee full traceability or retention enforcement unless off-chain events are somehow linked to their on-chain representations. Future improvements should therefore explore mechanisms such as cryptographic anchoring of off-chain log files or the use of trusted oracles to bridge this gap.

From a technical standpoint, the integration of blockchain into an archival system introduces considerable complexity. Setting up and maintaining a permissioned network, coordinating validator nodes, and managing smart contracts require specialized expertise that is not typically available in traditional IT departments. Moreover, such systems must be maintainable not only by developers but also understandable and operable by non-technical stakeholders. Without sufficient usability and organizational maturity, even technically sound solutions may not succeed in practice.

Key management represents another critical challenge. The current solution assumes that users securely handle their private keys and passphrases. While this model aligns with blockchain principles, it is often unrealistic in enterprise environments where staff turnover, role-based access control, and compliance obligations require structured key lifecycle management. Without robust tooling and defined procedures, private key handling becomes a single point of failure and undermines the overall trust model.

In addition to technical constraints, legal ambiguity persists. While blockchain can support the implementation of compliance mechanisms, it does not automatically ensure legal conformity. For instance, data stored on-chain cannot be deleted or modified. This immutability, while beneficial for integrity, may conflict with legal requirements for logical deletion or data minimization. Even though only hashed representations are stored, metadata such as UUIDs, public addresses, and retention information remain permanently accessible. The legal implications of such immutable records cannot be fully assessed within the scope of this work, and must be evaluated by legal experts or auditors in a real-world setting.

In summary, the findings indicate that blockchain technology is well-suited to support specific legal requirements in audit-proof archiving. Its effectiveness, however, depends on how carefully and consistently it is integrated into the overall system architecture. Blockchain should not be viewed as a replacement for conventional archival processes, but rather as a complementary trust layer that enhances integrity and transparency. In this role, it holds substantial potential, provided that its limitations are clearly understood and appropriately addressed in system design and organizational practice.

9.5. Limitations & Future Work

In addition to the limitations already mentioned, this section outlines further restrictions of the developed prototype, as well as potential approaches for future research and improvement. The developed system is currently in a prototype stage and therefore faces several technical and conceptual limitations that must be considered for productive deployment.

Limitations of Backend Integration for Read Operations In principle, all interactions with the blockchain should be executed from the backend to ensure system security. However, during the prototype implementation, a technical issue occurred regarding the execution of read operations ("view" functions) within the smart contract. The Go library used (geth) did not allow the execution of transactions that do not result in a state change (write operations) from the backend. Further investigation and modifications to the library did not resolve the issue. Testing the smart contract code in Remix[3] confirmed that the error did not originate from the contract itself. As a temporary solution, it was decided to execute read operations directly from the frontend using the Web3.js[4] library. The library successfully enabled data retrieval. It is important to note that read operations do not require sensitive data such as the user's private key for signing transactions. Therefore, this temporary shift to the frontend does not directly compromise the security of the system. Nevertheless, this architectural workaround should be addressed in future developments to maintain a consistent and secure system design.

Legal Assessment Although the prototype implements parts of the requirements defined in the GoBD and IDW RS FAIT 3, the legal assurance of audit-proof archiving through blockchain technology remains unresolved. Whether the system meets parts of the legal requirements for audit-proof archiving must ultimately be determined through a formal assessment by compliance experts or certified auditors.

Thus, the developed system does not yet constitute a complete audit-proof solution, but rather demonstrates how blockchain technology can support selected legal requirements in a meaningful way.

For productive deployment, the following requirements and functionalities should additionally be addressed and implemented:

- Creation of a comprehensive procedural documentation
- Provision of access for tax authorities (Z1, Z2, Z3)
- A robust key management
- Processes for the timely, complete, and accurate capture of documents
- Ensuring the authenticity of archived data
- Extension of access control mechanisms
- Expansion of logging and audit functionalities
- Ensuring the binding legal character
- Machine readability of the data

[3] Remix. https://remix.ethereum.org/
[4] Web3.js. https://web3js.readthedocs.io/en/v1.10.0/

- Measures to protect the availability of data
- Protection against data loss through backup and recovery strategies
- Implementation of an internal control system

Without these, the status of a conceptual prototype remains.

Blockchain Configuration and Performance Tests As part of the work, GoQuorum was selected in a configuration that is primarily designed for fast transaction processing. For this purpose, the block time was adjusted to two seconds. Although such short block times increase transaction throughput, they can also affect network traffic and network security. Systematic load and performance tests to evaluate the system under realistic conditions have not yet been carried out. Future research should assess the system's scalability and efficiency in various scenarios and explore alternative blockchain platforms or consensus mechanisms for potential improvements.

On- and Off-Chain Data Inconsistencies A key limitation of the system lies in the inability of the blockchain to verify the actual state of the Archive-Storage. The smart contract relies on status information provided by the backend, without the capability to independently validate it. For example, if an error occurs during a file upload, the smart contract may still assume that the file was successfully stored. This can lead to inconsistent system states in which a document is considered present on the blockchain, although it is not actually stored in the Archive-Storage. Future research could explore how suitable mechanisms for reporting off-chain events can be integrated into the system to improve the consistency between on-chain reference data and the actual status of off-chain documents.

Fine-grained Access Control The prototype implements a simple permission scheme in which the uploader manages access rights to the archived documents. Future work could explore a fine-grained, role-based access control model that extends the smart contracts and allows for greater flexibility and better alignment with internal organizational policies. In particular, it should be possible to define and dynamically adjust access permissions at an organizational level, rather than assigning them solely on a user basis.

UX improvements The SUS results indicate a lower level of user confidence and a perceived need for technical support. Future iterations should address these aspects by improving onboarding and providing clearer interface guidance. Step-by-step instructions or contextual explanations could help to reduce uncertainty and make the system more accessible.

10. Conclusion

This thesis aimed to examine to what extent blockchain technology can support the legal requirements of an audit-proof archiving system, with a focus on the regulatory frameworks applicable in Germany. The expected deliverables outlined at the beginning of this thesis were achieved through a four-step approach.

First, in Chapter 2, the term audit-proof archiving was initially defined, and a detailed overview of the relevant regulations and standards applicable in Germany was provided. Subsequently, the fundamentals and characteristics of blockchain technology were explained, highlighting both the advantages and limitations of its application in the context of record-keeping. The description of the legal criteria in chapter 2.2 provided an impression of the necessary requirements for an audit-proof archiving system. These were summarized in chapter 4 for the system to be developed. Second, in Chapter 5, various architectural approaches for a blockchain-based solution were explored. The choice fell on a hybrid system architecture in which the blockchain is integrated into the archive system in the form of an integrity and verification layer. This decision was made in light of blockchain's limitations in storing large volumes of data. Based on the identified potential of blockchain technology, a system concept was then developed that defines four core Smart Contract functions designed to support legal requirements addressed in the sub-research questions of this thesis. Third, to demonstrate the technical feasibility of the proposed concept, a fully functioning prototype was implemented using the Ethereum-based client GoQuorum, smart contracts, and traditional off-chain components as described in Chapter 7. The successful applicability of the implemented system was demonstrated in Chapter 8 through three practical use cases. Finally, in Chapter 9, the system was evaluated with respect to the research questions. In addition, user experience and usage costs were examined, and the system was assessed in terms of feasibility, limitations, and architectural implications. The findings of this thesis can be summarized in the following contributions:

- A structured requirements analysis for audit-proof electronic archiving based on German legal standards.
- The development of a hybrid system concept where blockchain technology enforces integrity, access control, retention, and traceability functions.
- The implementation of a prototype demonstrating practical feasibility for document lifecycle operations such as document upload, access, verification, and deletion.
- A critical evaluation of blockchain's strengths and limitations in supporting the compliance requirements for audit-proof archiving.

While blockchain technology cannot fulfill all requirements of audit-proof archiving on its own, it offers substantial benefits in key areas. However, due to current legal frameworks and technical limitations, blockchain cannot replace conventional archiving infrastructure, but can act as a verifiable trust layer.

With regard to the sub-questions, the findings show that the requirement of data integrity is reliably fulfilled through the use of cryptographic hashes stored immutably on the blockchain. The blockchain serves as an immutable source of truth, ensuring that all changes to archived documents can be reliably detected through comparison with on-chain reference data. Secure access is supported through smart contract based authorization, which transparently enforces access rights and prevents unauthorized usage. Confidentiality, however, is only partially supported, as it depends on external encryption mechanisms and cannot be guaranteed by blockchain technology alone. Retention enforcement is achieved at the application level via smart contracts, which prevent premature deletion of documents. Nevertheless, final deletion remains under the control of the off-chain storage infrastructure, limiting the overall enforceability. Lastly, traceability and verifiability are strengthened through immutable, time-stamped logging of on-chain interactions, offering an advantage over conventional systems that rely on mutable log files. However, actions outside the blockchain remain unrecorded unless integrated through additional mechanisms such as log anchoring or trusted oracles.

In summary, this thesis has demonstrated that blockchain technology can make a meaningful contribution to the legal requirements of audit-proof archiving by strengthening data integrity, authorization, traceability, and retention enforcement. The implemented prototype confirms this potential in a practical scenario while also highlighting current architectural and legal limitations. Since not all compliance requirements were addressed within the scope of this work, the system cannot yet be considered a complete solution for audit-proof archiving. Nevertheless, the results provide a concrete foundation for further development and contribute to the ongoing discussion about the practical use of blockchain in legally regulated environments.

Bibliography

[ABB+18] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference*, pages 1–15, 2018.

[AKAK21] Omar Alfandi, Salam Khanji, Liza Ahmad, and Asad Khattak. A survey on boosting iot security and privacy through blockchain. *Cluster Computing*, 24(1):37–55, 2021.

[AM+17] Ittai Abraham, Dahlia Malkhi, et al. The blockchain consensus layer and bft. *Bulletin of EATCS*, 3(123), 2017.

[AMP] Aanchal Anand, Matthew McKibbin, and Frank Pichel. Colored coins: Bitcoin, blockchain, and land administration colored coins: Bitcoin, blockchain, and land administration.

[B+96] John Brooke et al. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.

[B+14] Vitalik Buterin et al. A next-generation smart contract and decentralized application platform. *white paper*, 3(37):2–1, 2014.

[Bar20] Carol M Barnum. *Usability testing essentials: Ready, set... test!* Morgan Kaufmann, 2020.

[BBSBBS21] Alexander Burger, Sabine Burger-Stieber, Alexander Burger, and Sabine Burger-Stieber. Rechtliche grundlagen. *Grundlagen der Buchführung: Eine praxisorientierte Einführung mit Übungsaufgaben und Musterlösungen*, pages 7–27, 2021.

[Ben14] Juan Benet. Ipfs-content addressed, versioned, p2p file system. *arXiv preprint arXiv:1407.3561*, 2014.

[BI] Telekommunikation und neue Medien e.V Bundesverband Informationswirtschaft. Gobd-checkliste für dokumentenmanagement-systeme. `https://www.bitkom.org/sites/main/files/2021-03/210325_lf_gobd_checkliste_2.1.pdf`. Viewed on 2023-10-15.

[BKM08] Aaron Bangor, Philip T Kortum, and James T Miller. An empirical evaluation of the system usability scale. *Intl. Journal of Human–Computer Interaction*, 24(6):574–594, 2008.

[BL24] Michael Bissinger and Elisa Lutz. Historie der gobd. In *Verfahrensdokumentation nach GoBD*, pages 19–32. Springer, 2024.

[BLS+20] Eranga Bandara, Xueping Liang, Sachin Shetty, Wee Keong Ng, Peter Foytik, Nalin Ranasinghe, Kasun De Zoysa, Bård Langöy, and David Larsson. Lekana-blockchain based archive storage for large-scale cloud systems. In *International Conference on Blockchain*, pages 169–184. Springer, 2020.

[BRSS17] Thomas Bocek, Bruno B Rodrigues, Tim Strasser, and Burkhard Stiller. Blockchains everywhere-a use-case of blockchains in the pharma supply-chain. In *2017 IFIP/IEEE symposium on integrated network and service management (IM)*, pages 772–777. IEEE, 2017.

[Bun] Deutscher Steuerberaterverband e.V. Bundessteuerberaterkammer. Musterverfahrensdokumentation zum ersetzenden scannen. `https://www.bstbk.de/downloads/bstbk/steuerrecht-und-rechnungslegung/fachinfos/BStBK_Muster-VerfD-ersetzendes-Scannen_v2.0-2019-11-29.pdf`. Viewed on 2024-11-15.

[BW19] Danielle Alves Batista and Tim Weingaertner. Archcontract: using smart contracts for disposition. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 3060–3065. IEEE, 2019.

[CCK+18] Mohammad Jabed Morshed Chowdhury, Alan Colman, Muhammad Ashad Kabir, Jun Han, and Paul Sarda. Blockchain versus database: A critical analysis. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 1348–1353, 2018.

[CD16] Gertrude Chavez-Dreyfuss. Sweden tests blockchain technology for land registry. *Reuters, June*, 16, 2016.

[Con09] Project Consult. Revisionssichere archivierung versus rechtssichere archivierung. `https://www.project-consult.com/in_der_diskussion/revisionssichere-archivierung-versus-rechtssichere-archivierung/`, 2009. Viewed on 2022-11-28.

[CP22] Vittorio Capocasale and Guido Perboli. Standardizing smart contracts. *IEEE Access*, 10:91203–91212, 2022.

[CSR+17] Si Chen, Rui Shi, Zhuangyu Ren, Jiaqi Yan, Yani Shi, and Jinyu Zhang. A blockchain-based supply chain quality management framework. In *2017*

IEEE 14th International Conference on e-Business Engineering (ICEBE), pages 172–176. IEEE, 2017.

[Däs14] Sebastian Däs. *Compliance-konforme Einbindung biometrischer Authentifizierungssysteme in das betriebliche IT-Sicherheitsmanagement*. Springer, 2014.

[DD21] Christoph Deiminger and Christoph Deiminger. Unterstützung der corporate governance durch technologische neuerungen. *Unternehmensberichterstattung und technologischer Wandel: Eine Analyse von Einfluss-und Entwicklungspotentialen*, pages 253–328, 2021.

[dF] Bundesministerium der Finanzen. Grundsätze zur ordnungsmäßigen führung und aufbewahrung von büchern, aufzeichnungen und unterlagen in elektronischer form sowie zum datenzugriff (gobd). `https://ao.bundesfinanzministerium.de/ao/2023/Anhaenge/BMF-Schreiben-und-gleichlautende-Laendererlasse/Anhang-64/inhalt.html`. Viewed on 2023-10-21.

[dFMMR18] Damiano di Francesco Maesa, Paolo Mori, and Laura Ricci. Blockchain based access control services. *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 1379–1386, 2018.

[DKTA20] Mohammad Dabbagh, Mohsen Kakavand, Mohammad Tahir, and Angela Amphawan. Performance analysis of blockchain platforms: Empirical evaluation of hyperledger fabric and ethereum. *2020 IEEE 2nd International Conference on Artificial Intelligence in Engineering and Technology (IICAIET)*, pages 1–6, 2020.

[DPOB22] Trinh Viet Doan, Yiannis Psaras, Jörg Ott, and Vaibhav Bajpai. Toward decentralized cloud storage with ipfs: Opportunities, challenges, and future considerations. *IEEE Internet Computing*, 26(6):7–15, 2022.

[DR12] Luciana Duranti and Corinne Rogers. Trust in digital records: An increasingly cloudy legal area. *Computer Law & Security Review*, 28(5), 2012.

[DS20] Faiq Dzulfikar and Ajib Susanto. Implementation of smart contracts ethereum blockchain in web-based electronic voting (e-voting). *Jurnal Transformatika*, 18:56–62, 07 2020.

[DT22] Erik Daniel and Florian Tschorsch. Ipfs and friends: A qualitative comparison of next generation peer-to-peer data networks. *IEEE Communications Surveys Tutorials*, 24(1):31–52, 2022.

[dWFfI06] Institut der Wirtschaftsprüfer Fachausschuss für Informationstechnologie. Idw-rs-fait-3 - idw stellungnahme zur rechnungslegung: Grundsätze ordnungsmäßiger buchführung beim einsatz elektronischer archivierungsverfahren. http://www.weg-mit-dem-papier.de/files/pdf/FAIT3.PDF, 2006. Viewed on 2023-10-19.

[DYA+23] Rupesh Kumar Dewang, Mahendra Pratap Yadav, Surbhit Awasthi, Om Raj, Arvind Mewada, and Kamlakant Laxman Bawankule. Data secure application: An application that allows developers to store user data securely using blockchain and ipfs. *Multimedia Tools and Applications*, pages 1–27, 2023.

[etha] Danksharding. https://ethereum.org/en/roadmap/danksharding/. Viewed on 2023-12-21.

[ethb] Ethereum roadmap. https://ethereum.org/en/roadmap/#what-about-sharding/. Viewed on 2023-12-21.

[ethc] Proof-of-stake (pos). https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/. Viewed on 2022-11-28.

[fSidI19] Bundesamt für Sicherheit in der Informationstechnik. Bsi technische richtlinie 03125 beweiswerterhaltung kryptographisch signierter dokumente. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR03125/BSI_TR_03125_V1_2_2.pdf?__blob=publicationFile&v=5, 2019. Viewed on 2023-10-02.

[fSidI24] Bundesamt für Sicherheit in der Informationstechnik. Technische richtlinie bsi tr-02102-1 kryptographische verfahren: Empfehlungen und schlüssellängen. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102.pdf?__blob=publicationFile&v=10, 2024. Viewed on 2024-03-05.

[fwVe] AWV – Arbeitsgemeinschaft für wirtschaftliche Verwaltung e.V. Gobd ein praxisleitfaden für unternehmen, version 2.2. https://www.awv-net.de. Viewed on 2024-11-09.

[GB17] Wolfgang Heinrich Stefan Groß and Thorsten Brand. Die gobd in der praxis-ein leitfaden für die unternehmenspraxis, 2017.

[GBD23] Mongetro Goint, Cyrille Bertelle, and Claude Duvallet. Secure access control to data in off-chain storage in blockchain-based consent systems. *Mathematics*, 11(7):1592, 2023.

[goe] goethereumbook.org. Events. https://goethereumbook.org/events/. Viewed on 2023-11-12.

[HCK17] Seyoung Huh, Sangrae Cho, and Soohyung Kim. Managing iot devices using blockchain platform. In *2017 19th international conference on advanced communication technology (ICACT)*, pages 464–467. IEEE, 2017.

[HHT+22] Jialu Hao, Cheng Huang, Wenjuan Tang, Yang Zhang, and Shuai Yuan. Smart contract-based access control through off-chain signature and on-chain evaluation. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 69:2221–2225, 2022.

[HKG16] Jannis Holthusen, Simon Kufeld, and Florian Glatz. Blockchain-technologie - revisionssichere archivierung. *Online verfügbar: https://www2.deloitte.com/content/dam/Deloitte/de/Documents/Innovation/Blockchain_Revisionss*

[HSE+18] Thomas Hepp, Matthew Sharinghousen, Philip Ehret, Alexander Schoenhals, and Bela Gipp. On-chain vs. off-chain storage for supply-and blockchain integration. *it-Information Technology*, 60(5-6):283–291, 2018.

[HWRBS17] Frank Hofmann, Simone Wurster, Eyal Ron, and Moritz Böhmecke-Schwafert. The immutability concept of blockchains and benefits of early standardization. In *2017 ITU Kaleidoscope: Challenges for a Data-Driven Society (ITU K)*, pages 1–8, 2017.

[IPF] Ipfs docs. `https://docs.ipfs.tech/`. Viewed on 2023-12-15.

[ITL+22] Aisyah Ismail, Mark Toohey, Young Choon Lee, Zhongli Dong, and Albert Y. Zomaya. Cost and performance analysis on decentralized file systems for blockchain-based applications: State-of-the-art report. In *2022 IEEE International Conference on Blockchain (Blockchain)*, pages 230–237, 2022.

[JHW18] Archana Prashanth Joshi, Meng Han, and Yan Wang. A survey on security and privacy issues of blockchain technology. *Mathematical foundations of computing*, 1(2), 2018.

[Jos21] Shashank Joshi. Feasibility of proof of authority as a consensus protocol model. *arXiv preprint arXiv:2109.02480*, 2021.

[Kam02] Ulrich Kampffmeyer. Revisionssichere Archivierung. `http://www.contentmanager.de/magazin/artikel_221_revisionssichere_archivierung.html`, contentmanager.de, September 2002. Artikel über Revisionssicherheit.

[Kam03] Ulrich Kampffmeyer. Elektronische archivierung und storage-technologien, 2003.

[Kam12] Ulrich Kampffmeyer. Elektronische archivierung, 2012.

[KB18] Rosco Kalis and Adam Belloum. Validating data integrity with blockchain. In *2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 272–277, 2018.

[KD08] Michael Klotz and Dietrich-W Dorn. It-compliance—begriff, umfang und relevante regelwerke. *HMD Praxis der Wirtschaftsinformatik*, 45:5–14, 2008.

[Kei17] Garret Keirns. Blockchain land registry tech gets test in brazil. *CoinDesk News*, 5:2017, 2017.

[KHD17] Kari Korpela, Jukka Hallikas, and Tomi Dahlberg. Digital supply chain transformation toward blockchain integration. In *proceedings of the 50th Hawaii international conference on system sciences*, 2017.

[KK23] Sunil Kumar and Dilip Kumar. Securing of cloud storage data using hybrid aes-ecc cryptographic approach. *Journal of Mobile Multimedia*, pages 363–388, 2023.

[KLGG+21] Shafaq Naheed Khan, Faiza Loukil, Chirine Ghedira-Guegan, Elhadj Benkhelifa, and Anoud Bani-Hani. Blockchain smart contracts: Applications, challenges, and future trends. *Peer-to-peer Networking and Applications*, 14:2901–2925, 2021.

[Kli20] Phillip Kling. *GoBD*. IDW Verlag GmbH, 2020.

[KM97] Dr Ulrich Kampffmeyer and Barbara Merkel. *Grundsätze der elektronischen Archivierung in Deutschland*. VOI, 1997.

[KSKS21] Tomasz Kusber, Steffen Schwalm, Ulrike Korte, and Kalinda Schamburger. Records management and long-term preservation of evidence in dlt. 2021.

[Lee20] Julia Leeder. *Grundsätze zur ordnungsgemäßen Führung und Aufbewahrung von Büchern, Aufzeichnungen und Unterlagen in elektronischer Form, sowie zum Datenzugriff (GoBD)*. PhD thesis, 2020.

[Lem] Investigator Dr Victoria L Lemieux. Help or hype?

[Lem16] Victoria Louise Lemieux. Trusting records: is blockchain technology the answer? *Records Management Journal*, 2016.

[Lem17a] Victoria L Lemieux. Blockchain and distributed ledgers as trusted recordkeeping systems. In *Future technologies conference (FTC)*, volume 2017, 2017.

[Lem17b] Victoria L Lemieux. A typology of blockchain recordkeeping solutions and some reflections on their implications for the future of archival preservation. In *2017 IEEE international conference on big data (Big Data)*, pages 2271–2278. IEEE, 2017.

[LHBJ19] Victoria Lemieux, Darra Hofman, Danielle Batista, and Alysha Joo. Blockchain technology & recordkeeping. *ARMA International Educational Foundation*, 2019.

[LJW+19] Jiazhuo Lyu, Zoe L Jiang, Xuan Wang, Zhenhao Nong, Man Ho Au, and Junbin Fang. A secure decentralized trustless e-voting system based on smart contract. In *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 570–577. IEEE, 2019.

[LL17] Iuon-Chang Lin and Tzu-Chun Liao. A survey of blockchain security issues and challenges. *Int. J. Netw. Secur.*, 19(5):653–659, 2017.

[LND19] Nino Lazuashvili, Alex Norta, and Dirk Draheim. Integration of blockchain technology into a land registration system for immutable traceability: a casestudy of georgia. In *International Conference on Business Process Management*, pages 219–233. Springer, 2019.

[Mey22] Katrin Meyer. Blockchain: Was ist das? eigenschaften und anwendungen. *FMS-BERICHTE*, 2022.

[MSH17] Patrick McCorry, Siamak F Shahandashti, and Feng Hao. A smart contract for boardroom voting with maximum voter privacy. In *International conference on financial cryptography and data security*, pages 357–375. Springer, 2017.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[NNJ19] M Niranjanamurthy, BN Nithya, and SJCC Jagannatha. Analysis of blockchain technology: pros, cons and swot. *Cluster Computing*, 22(6):14743–14757, 2019.

[Nzu19] Silas Nzuva. Smart contracts implementation, applications, benefits, and limitations. *School of Computing and Information Technology, Jomo Kenyatta University of Agriculture and Technology, Nairobi, Kenya*, 2019.

[OAEAO16] Aafaf Ouaddah, Anas Abou Elkalam, and Abdellah Ait Ouahman. Fairaccess: a new blockchain-based access control framework for the internet of things. *Security and communication networks*, 9(18):5943–5964, 2016.

[OEO17] Aafaf Ouaddah, Anas Abou Elkalam, and Abdellah Ait Ouahman. Towards a novel privacy-preserving access control model based on blockchain technology in iot. In *Europe and MENA cooperation advances in information and communication technologies*, pages 523–533. Springer, 2017.

[Org09] VOI-Verband Organisations. und informationssysteme e. *V. Merksätze des VOI zur revisionssicheren elektronischen Archivierung*, page 25, 2009.

[PMTM22] Marie-Christin Papen, Katrin Marques Magalhaes, Sebastian Tenbrock, and Christian Märkel. Digitalisierungsanwendungen und identifikation von digitalisierungstrends im mittelstand. Technical report, WIK Diskussionsbeitrag, 2022.

[RBR20] Marco Rosa, Joao Paulo Barraca, and Nelson Pacheco Rocha. Blockchain structures to guarantee logging integrity of a digital platform to support community-dwelling older adults. *Cluster Computing*, 23(3):1887–1898, 2020.

[RG15] Johannes Riepolt and Stephan Greulich. *Das BMF-Schreiben zu den GoBD: neue Grundsätze zur elektronischen Buchführung und zum Datenzugriff*. DATEV eG, 2015.

[RGG+18] Michel Rauchs, Andrew Glidden, Brian Gordon, Gina C Pieters, Martino Recanatini, François Rostand, Kathryn Vagneur, and Bryan Zheng Zhang. Distributed ledger technology systems: A conceptual framework. *Available at SSRN 3230013*, 2018.

[RMK18] Thomas Renner, Johannes Müller, and Odej Kao. Endolith: A blockchain-based framework to enhance data retention in cloud storages. In *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pages 627–634, 2018.

[SB21] Hrvoje Stančić and Vladimir Bralić. Digital archives relying on blockchain: overcoming the limitations of data immutability. *Computers*, 10(8):91, 2021.

[SC17] Ankit Songara and Lokesh Chouhan. Blockchain: a decentralized technique for securing internet of things. In *Conference on Emerging Trends in Engineering Innovations & Technology Management (ICET: EITM-2017)*, 2017.

[SJB19] Pratima Sharma, Rajni Jindal, and Malaya Dutta Borah. Blockchain-based integrity protection system for cloud storage. *2019 4th Technology Innovation Management and Engineering Science International Conference (TIMES-iCON)*, pages 1–5, 2019.

[sol] Solidity. `https://docs.soliditylang.org/en/v0.8.17/`. Viewed on 2022-11-28.

[SY19] Mahendra Kumar Shrivas and Thomas Yeboah. The disruptive blockchain: types, platforms and applications. *Texila International Journal of Academic Research*, 3:17–39, 2019.

[SZJS22] Reza Soltani, Marzia Zaman, Rohit Joshi, and Srinivas Sampalli. Distributed ledger technologies and their applications: A review. *Applied Sciences*, 12(15):7898, 2022.

[TA13] Tom Tullis and Bill Albert. Chapter 6 - self-reported metrics. In Tom Tullis and Bill Albert, editors, *Measuring the User Experience (Second Edition)*, Interactive Technologies, pages 121–161. Morgan Kaufmann, Boston, second edition edition, 2013.

[THLG18] Choon Beng Tan, Mohd Hanafi Ahmad Hijazi, Yuto Lim, and Abdullah Gani. A survey on proof of retrievability for cloud data integrity and availability: Cloud storage state-of-the-art, issues, solutions and future trends. *Journal of Network and Computer Applications*, 110:75–86, 2018.

[TS+04] Thomas S Tullis, Jacqueline N Stetson, et al. A comparison of questionnaires for assessing website usability. In *Usability professional association conference*, volume 1, pages 1–12. Minneapolis, USA, 2004.

[UK97] Jörg Rogalla Ulrich Kampffmeyer. *Grundsätze der elektronischen Archivierung*. 1997.

[WBKV21] Tim Weingärtner, Danielle Batista, Sandro Köchli, and Gilles Voutat. Prototyping a smart contract based public procurement to fight corruption. *Computers*, 10(7):85, 2021.

[WEKJ17] Christian Welzel, Klaus-Peter Eckert, Fabian Kirstein, and Volker Jacumeit. *Mythos Blockchain: Herausforderung für den öffentlichen Sektor*. Kompetenzzentrum Öffentliche IT, Fraunhofer-Institut für Offene ..., 2017.

[WG18a] Karl Wüst and Arthur Gervais. Do you need a blockchain? In *2018 crypto valley conference on blockchain technology (CVCBT)*, pages 45–54. IEEE, 2018.

[WG18b] Karl Wüst and Arthur Gervais. Do you need a blockchain? In *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 45–54, 2018.

[WYW+18] Shuai Wang, Yong Yuan, Xiao Wang, Juanjuan Li, Rui Qin, and Fei-Yue Wang. An overview of smart contract: Architecture, applications, and future trends. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 108–113, 2018.

[WZRM21] Maximilian Wöhrer, Uwe Zdun, and Stefanie Rinderle-Ma. Architecture design of blockchain-based applications. In *2021 3rd Conference on Blockchain Research Applications for Innovative Networks and Services (BRAINS)*, pages 173–180, 2021.

[XCW+22] Huanliang Xiong, Muxi Chen, Canghai Wu, Yingding Zhao, and Wenlong Yi. Research on progress of blockchain consensus algorithm: A review on recent progress of blockchain consensus algorithms. *Future Internet*, 14(2):47, 2022.

[XSA+17] QI Xia, Emmanuel Boateng Sifah, Kwame Omono Asamoah, Jianbin Gao, Xiaojiang Du, and Mohsen Guizani. Medshare: Trust-less medical data sharing among cloud service providers via blockchain. *IEEE access*, 5:14757–14767, 2017.

[XSS+17] Qi Xia, Emmanuel Boateng Sifah, Abla Smahi, Sandro Amofa, and Xiaosong Zhang. Bbds: Blockchain-based data sharing for electronic medical records in cloud environments. *Information*, 8(2):44, 2017.

[XYH18] Yang Xinyi, Zhang Yi, and Yulin He. Technical characteristics and model of blockchain. In *2018 10th International Conference on Communication Software and Networks (ICCSN)*, pages 562–566, 2018.

[YCC19] Wei-Kai Yang, Jie-Si Chen, and Yeong-Sheng Chen. An electronic medical record management system based on smart contracts. In *2019 Twelfth International Conference on Ubi-Media Computing (Ubi-Media)*, pages 220–223, 2019.

[YNZ+19] Kazuhiro Yamashita, Yoshihide Nomura, Ence Zhou, Bingfeng Pi, and Sun Jun. Potential risks of hyperledger fabric smart contracts. *2019 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*, pages 1–10, 2019.

[YSJAH22] Ibrar Yaqoob, Khaled Salah, Raja Jayaraman, and Yousof Al-Hammadi. Blockchain for healthcare data management: opportunities, challenges, and future recommendations. *Neural Computing and Applications*, pages 1–16, 2022.

[YYNH18] Xuechao Yang, Xun Yi, Surya Nepal, and Fengling Han. Decentralized voting: a self-tallying voting system using a smart contract on the ethereum blockchain. In *International Conference on Web Information Systems Engineering*, pages 18–35. Springer, 2018.

A. Appendix

The repository of the developed prototype "ArchiveChain" for this thesis can be accessed at: https://git01lab.cs.univie.ac.at/marcialf95/archive . It contains the complete source code together with a Postman collection, which is provided for the Share Access use case described in Section 8.2.1.