APPLYING BLOCKCHAIN TECHNOLOGY TO ASPECTS OF ELECTRONIC HEALTH RECORDS IN SOUTH AFRICA: LESSONS LEARNT

R. ADLAM

2020

APPLYING BLOCKCHAIN TECHNOLOGY TO ASPECTS OF ELECTRONIC HEALTH RECORDS IN SOUTH AFRICA: LESSONS LEARNT

By

Ryno Adlam

Submitted in fulfilment of the requirements for the degree of Master of Information Technology to be awarded at the Nelson Mandela University

April 2020

Supervisor: Dr. Bertram Haskins

NELSON MANDELA UNIVERSITY

DECLARATION BY CANDIDATE

NAME: Ryno Adlam

STUDENT NUMBER: 213397609

QUALIFICATION: Master of Information Technology

TITLE OF PROJECT: Applying blockchain technology to aspects of electronic health records in South Africa: Lessons learnt

DECLARATION:

In accordance with Rule G5.6.3, I hereby declare that the above-mentioned treatise/ dissertation/ thesis is my own work and that it has not previously been submitted for assessment to another University or for another qualification.

SIGNATURE: 

DATE: 4 December 2019

Abstract

The purpose of this study was to explore the applicability of blockchain technology as a viable alternative for the secure storage and distribution of electronic health records in a South African context. The adoption of electronic health records (EHRs) has grown over recent years. Electronic health records (EHRs) can be seen as electronic versions of patients' medical history. EHRs promise benefits such as improving the quality of care, reducing medical errors, reducing costs, saving time, and enhancing the availability and sharing of medical records.

Blockchain, in simple terms, could be seen as a distributed database controlled by a group of individuals. Blockchain technology differs from other distributed ledger technology by bundling unrelated data into blocks that are chained together in a linked-list manner, hence the name blockchain. Blockchain technology strives to provide desirable features, such as decentralization, immutability, audibility, and transparency. EHRs are traditionally constructed with a cloud-based infrastructure to promote the storing and distribution of medical records. These medical records are commonly stored in a centralized architecture, such as a relational database. The centralized architecture employed by EHRs may present a single point of failure. These kinds of failures may lead to data-breaches. The cloud-based infrastructure is effective and efficient from an availability standpoint. The increased availability of electronic health records has brought forth challenges related to the security and privacy of the patient's medical records. The sensitive nature of EHRs attracts the attention of cyber-criminals. There has been a rise in the number of data breaches related to electronic health records. The traditional infrastructure used by electronic health records can no longer ensure the privacy and security of patient's medical records. To determine whether blockchain is a viable alternative to these approaches, the main objective of this study was to compile a technical report on the applicability of aspects of blockchain technology to the secure storage and distribution of electronic health records.

The study first conducted a literature review to gather background on the current state of electronic health records and blockchain technology. The results of the literature review were used to compile an initial report. Experiments were conducted with various aspects of blockchain technology to build a technical baseline and to ultimately validate the initial report. The insights gained from the experiments served to refine the initial report into a final technical report. The final deliverable of this study was to devise a technical report. The technical report serves as a generalized overview of the applicability of blockchain technology as a secure storage and distribution mechanism for electronic health records. The main topics covered by the technical report to outline the applicability of blockchain technology to EHRs are as follows: authentication, authorization, audit log, storage and transactions.

The insights gained from the study illustrate that permissioned blockchain technology can enhance the traditional AAA security scheme employed by traditional EHRs. The AAA security scheme entails the use of certificate-based authentication and attribute-based access control for authorization. Audit logs can be stored in a semi-decentralized architecture that can enhance the security and privacy of audit logs. Using blockchain technology for storing electronic health records might not be a viable alternative to traditional EHRs architecture. Blockchain technology violates certain privacy regulations as information is stored in a permanent manner. Furthermore, blockchain technology is not optimized for dealing with large volumes of data. However, blockchain technology could be used to store a cryptographic hash of electronic health records to ensure the integrity of records. Permissioned blockchain technology can enhance the EHRs transaction process by transacting health records in a peer-to-peer infrastructure. In doing so, the above-mentioned AAA security scheme can enhance the security, confidentiality, and integrity of electronic health records shared across organizational bounds.

Acknowledgements

I would firstly like to express the deepest appreciation to my supervisor and mentor, Dr. Bertram Haskins for his contentious guidance and support throughout this research process. Without his assistance and leading hand, this study would not have been possible.

Thank you to both my parents, Frank and Madeleine Adlam. I am greatly thankful for my upbringing with love and encouragement leading up to this stage. They have raised me to be determined, passionate, courageous and hard-working. No words are enough to express my gratitude for all that you have done for me and continue doing. A further thank you to everyone else that may have played an important part in my journey to completing this master's dissertation.

Table of Contents

List of Figures

List of Tables

List of Code snippets

Chapter 1

Introduction

The blockchain revolution is here. This study will focus on how blockchain technology could be used or adapted to solve some of the challenges that the health care industry currently face. Health care institutions traditionally use a client-server model to maintain their medical records. These medical records are generally stored in a centralized architecture, such as a relational database, which represents a single point of failure (Liang, Zhao, Shetty, Liu, & Li, 2017). Health records are often tampered with for various reasons, namely insurance coverage, criminal offenses and more (Sharpe, 1999). Patients may receive care at multiple institutions throughout their lives. Health care institutions are required by law to record patients' medical and personal particulars at the first consultation. Patients are required to provide their medical records from other institutions, upon request. This process is time-consuming and could also bring about gaps in the patients' medical history.

Health care institutions are considered to be the owners of the medical records they store and, as such, are responsible for complying with regulations, such as the Protection of Personal Information Act, 2013 (Act No 4 of 2013) to ensure the confidentiality and integrity of the records (Katurura & Cilliers, 2016). Therefore, health care institutions are reluctant to share medical records for obvious reasons, such as privacy concerns ("A review of cross organizational healthcare data sharing", 2015). This results in patients' medical records being fragmented across multiple institutions, which, in turn, could result in patients being misdiagnosed and/or receiving incorrect treatment/prescriptions (Ekblaw, Azaria, Halamka, & Lippman, 2016). The health care industry could benefit from the characteristics of blockchain technology, such as decentralisation, immutability and audibility. Blockchain technology, therefore, has the potential of enhancing various aspects of the health care industry (Catalini, 2017).

1.1 Problem statement

Electronic health records are commonly stored and distributed in a way that represents a single point of failure and ownership, while having to adhere to specific privacy and storage requirements.

1.2 Thesis statement

Aspects of blockchain technology present a viable alternative for the secure storage and distribution of electronic health records.

1.3 Research objective

1.3.1 Primary objective

The problem statement will be addressed by completing the following primary research objective: *To compile a technical report on the applicability of aspects of blockchain technology to the secure storage and distribution of electronic health records.*

1.3.2 Secondary objectives

In support of the primary research objective, the secondary research objectives have been identified as follows:

1. *To identify the policies and regulations governing electronic health records in South Africa.*
2. *To determine which aspects of blockchain technologies are most suitable for the secure storage and distribution of information.*
3. *To conduct experiments with aspects of blockchain technologies, for the secure storage and distribution of electronic health records.*

1.4 Research design

The purpose of this study is to learn more about the applicability of blockchain technology as a secure storage and distribution mechanism for electronic health records in South Africa. Research design is the plan of action for a study, producing the overall framework

for collecting data (Kothari, 2004). Quantitative research is the process of using multiple sources to gather and analyze data in a structured manner. A quantitative research approach makes use of experimental methods along with quantitative measures to test a hypothesis (Golafshani, 2003). A research methodology is a systematic approach you follow from problem to solution (Kothari, 2004). A research methodology consists of various methods. For this study, the following methods will be used, namely literature review, experimentation, and inductive reasoning.

1.4.1 Literature review

According to Fink (2010), "A literature review is a systematic, explicit and reproducible method for identifying, evaluating and synthesizing the existing body of recorded work produced by researchers, scholars, and practitioners" (p. 1). This study will conduct a literature review on two topics. The first literature review topic will identify the policies and regulations governing electronic health records in South Africa, thereby meeting the first secondary objective 1. The second literature review topic will be used to contrast blockchain technologies suitable for storing and sharing information, thereby meeting the secondary objective 2.

1.4.2 Experimentation

Experimentation is the process of following a structured set of tests to validate a hypothesis (Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, 2012). In this study, experimentation will be used to evaluate the validity of an initial report compiled from the literature. Insights gained from the experiments conducted with various aspects of blockchain technologies will be used to refine the initial report. This will be done to satisfy the third secondary objective.

1.4.3 Inductive reasoning

Inductive reasoning is the processes of forming a specific observation, which leads to a generalized conclusion (Sauce & Matzel, 2017). In this study, inductive reasoning will be used to devise an initial report, based on the findings from the literature review. Inductive reasoning will also be used to incorporate the lessons learnt from the experiments conducted with various aspects of blockchain technologies into a final report. This report serves as the main deliverable of the study and addresses the main objective.

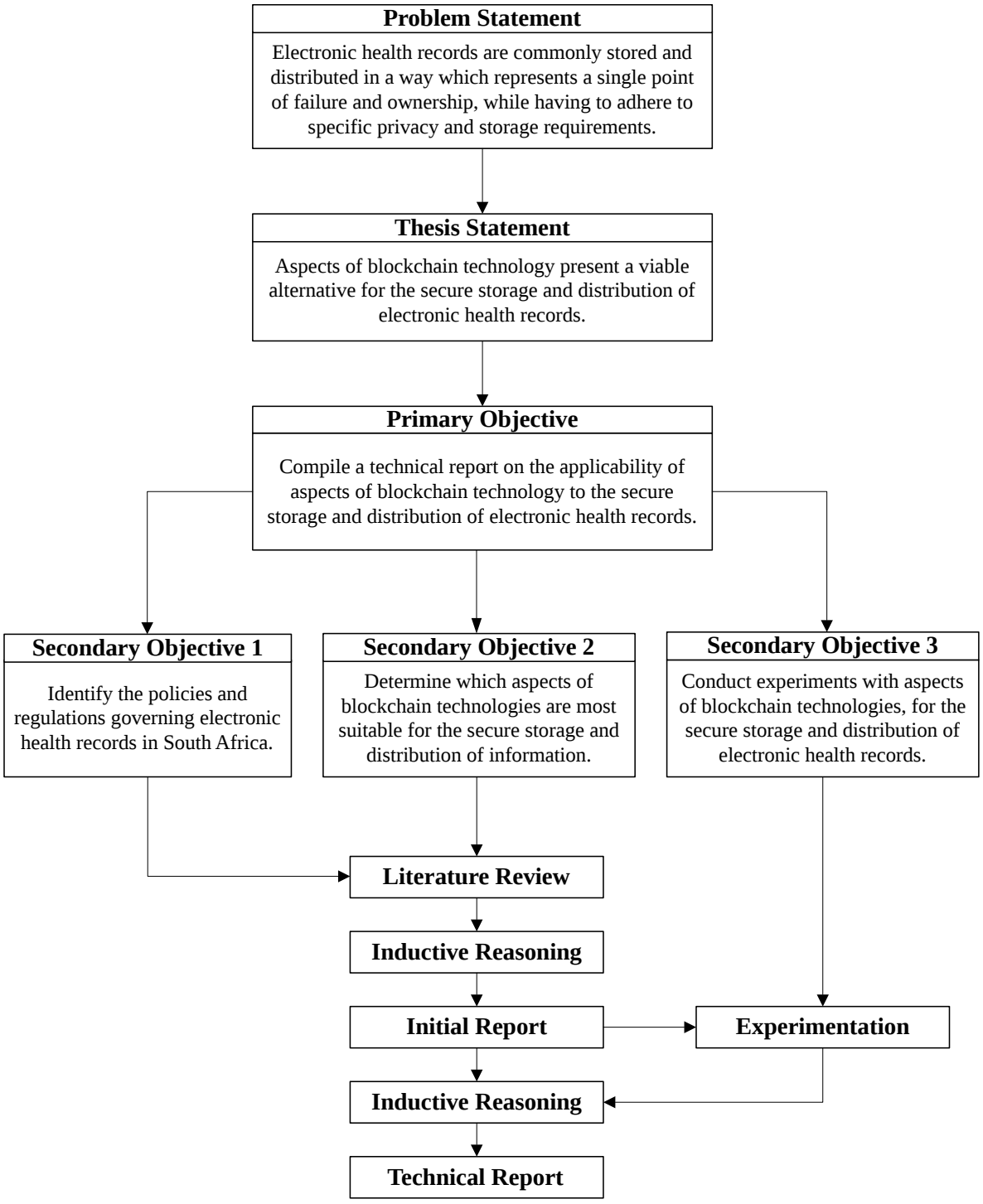

Figure 1.1: Research design diagram

1.5 Delineation

The prototype devised in this study will not be implemented at any institution. Although healthcare records are a global concept, this study will only focus on the requirements related to electronic health records in South Africa, regardless of any similarities to standards in other countries.

1.6 Chapter layout

This study will consist of six chapters which are graphically illustrated in Figure 1.2. The remainder of this section will provide a brief overview of each chapter. Chapter 1 serves as the introduction and will outline some of the ideas that will be covered in this study. In Chapter 2, a literature review will be conducted to identify the policies and regulations governing electronic health records in South Africa. Chapter 3 will present a literature review to contrast blockchain technologies suitable for the secure storage and distribution of information. Chapter 4 will present an initial report based on the findings of the literature review outlined in chapters 2 and 3. Chapter 5 will outline the results and insights gained from the experiments conducted with blockchain technology for the secure storage and distribution of electronic health records. The study will be concluded in Chapter 6.

1.7 Conclusion

This chapter serves as an overview of the purpose and content that will be covered in this study. The problem statement and thesis statement for this study were outlined to define the purpose of this study. The research objectives were defined to outline the goals of the study. The main objective of this study is to devise a technical report to satisfy the problem statement. The research design was discussed to illustrate how this study aims to achieve the defined objectives. The first half of the study will explore the current state of electronic health records and blockchain technology to gain a broader perspective of the topics. In doing so, inductive reasoning can be applied to devise an initial report, which could provide a focused view on how blockchain technology could improve electronic health records. The information outlined in the initial report will be used to conduct experiments with blockchain technology to enhance electronic health records. The insights and results gained from the experiments will be used to refine the initial report into a technical report. The technical report will serve as the main

deliverable of this study and will be attached to this document as an appendix. The concluding chapter of this study will evaluate the findings and discuss how the problem statement was addressed.

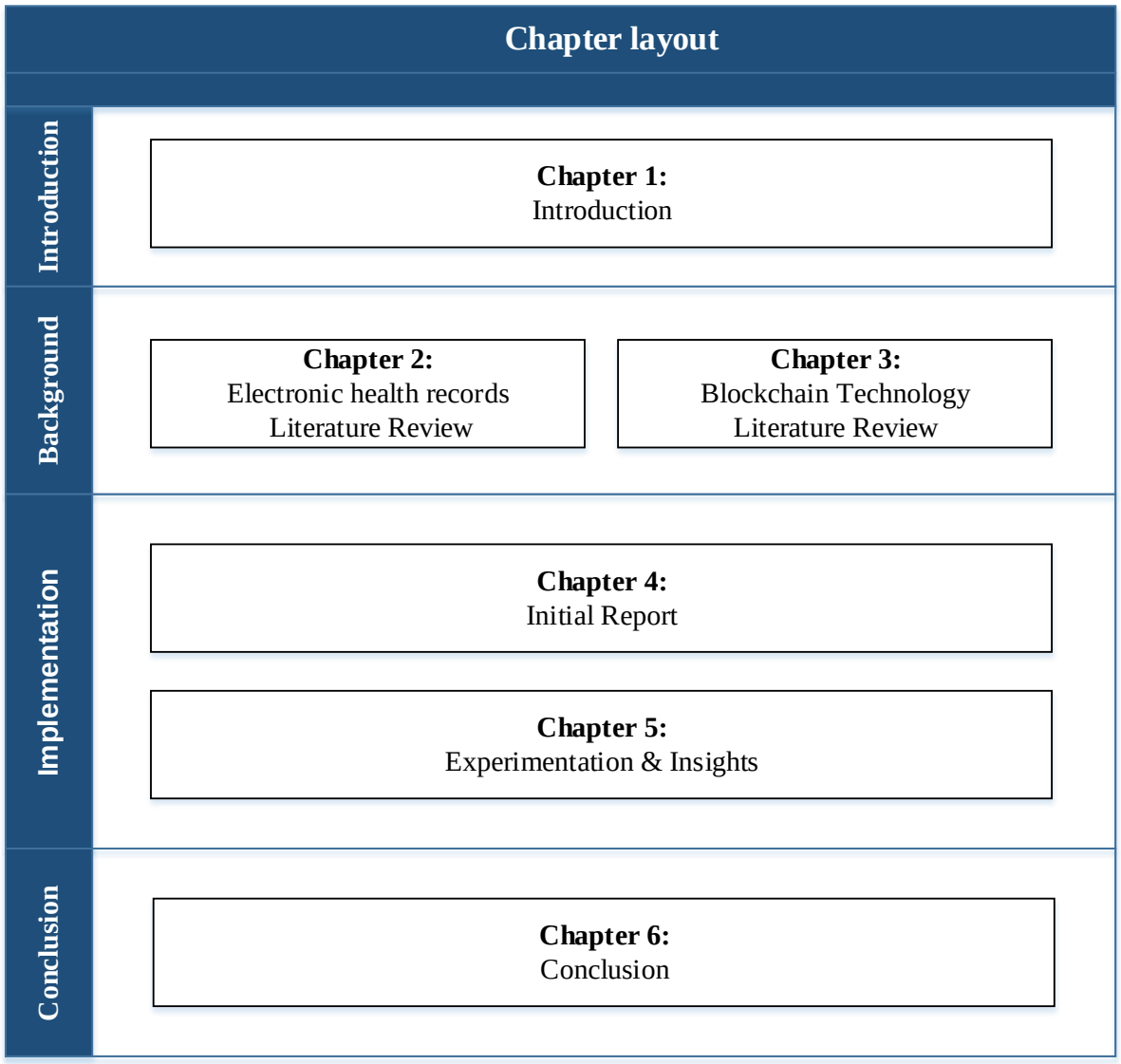

Figure 1.2: Chapter layout diagram

Chapter 2

Electronic health records

2.1 Introduction

There has been a rise in the adoption of electronic health records (EHRs) over recent years (Hutton, 2016). Electronic health records (EHRs) can be seen as electronic versions of patients' medical history (Dekker & Etalle, 2007). EHRs promise benefits such as improving the quality of care, reducing medical errors, reducing costs, saving time and enhancing the availability and sharing of medical records (Thakkar & Davis, 2006). As EHR systems deal with highly sensitive information such as patients' demographics, diagnoses, vital signs, past medical history, etc, they are subject to strict privacy regulations, such as the PoPI Act of South Africa (Katurura & Cilliers, 2016). EHR systems are generally implemented with a client-server model that provides access to health records through web and/or mobile interface but are currently facing several challenges, such as data breaches, privacy compromises, interoperability, audibility, and fraud. EHR systems currently make use of a centralized architecture that requires a centralized authority of trust and leaves medical records vulnerable to a single point of failure (Liang et al., 2017). Blockchain technology has the potential to institutionalize secure data exchange in the healthcare industry, which could benefit from features such as decentralization, immutability, audibility and transparency (Catalini, 2017). This, in turn, could lead to a cost-effective, time-saving and simpler infrastructure compared to conventional public key infrastructure (PKI) networks (Bashir, 2017, p. 438). This chapter serves to address secondary objective one which aims to identify the policies and regulations governing electronic health records in South Africa.

2.2 Challenges of traditional EHR systems

Electronic health records contain highly sensitive information and as a result, face constant cyberattacks (Kshetri, 2018). Recent reports illustrate that more than 112 million electronic health records were exposed through data breaches in 2015 (Kshetri, 2018). The number and severity of cyberattacks on electronic health records is increasing (Ronquillo, Winterholler, Cwikla, & Szymanski, 2018). Hackers consider electronic health records as a one-stop-shop for all their sensitive data needs. The stolen data is either sold on the black market or the hackers hold the electronic health records for ransom. The WannaCry ransomware cyberattack in 2017 affected countless healthcare providers who were forced to either pay the ransom or to close their doors to further patient care (Ronquillo et al., 2018). It is thus clear that the traditional infrastructure utilized by electronic health record systems cannot ensure the security and privacy of patients' health records (Kshetri, 2018). An overview of the current EHR model is illustrated in Figure 2.1.

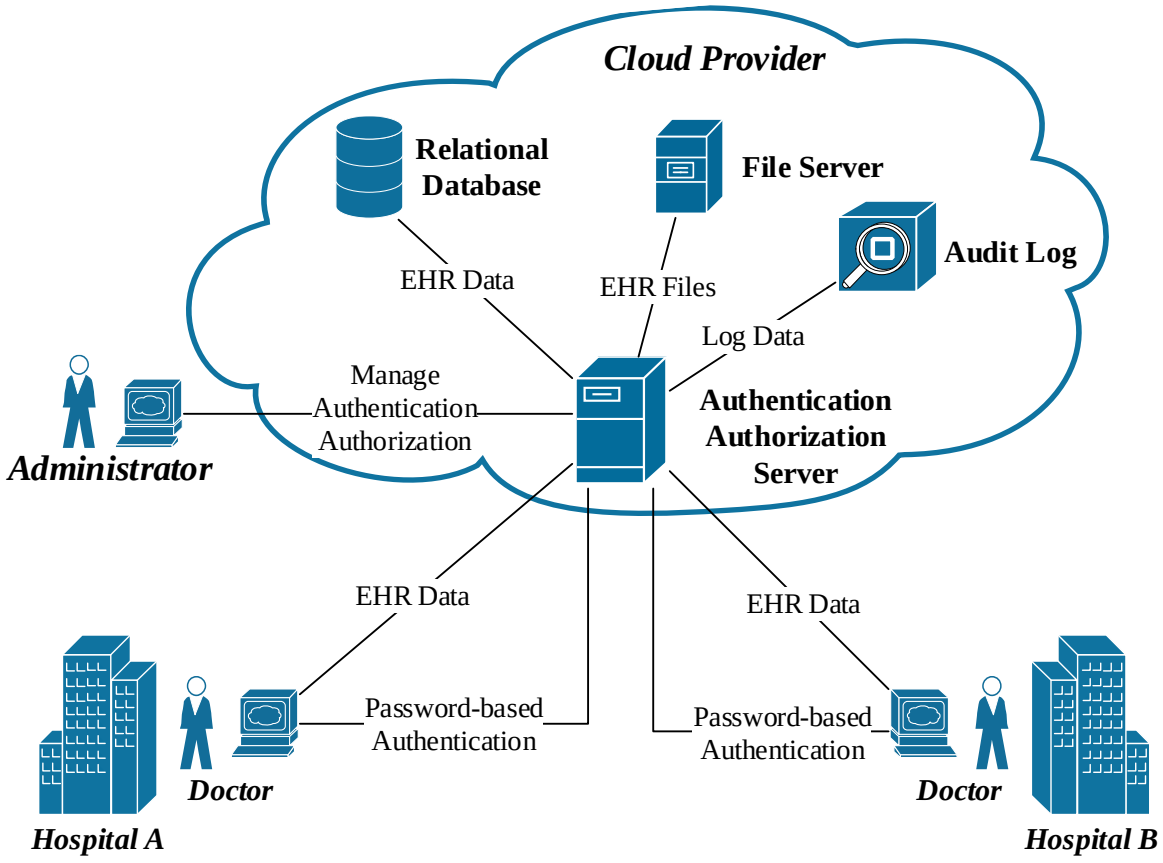

Figure 2.1: EHR system overview diagram

2.2.1 Authentication

Authentication is to identify a user requesting access to the system. Systems need to be able to identify users to restrict access to that system's functions. Users' identities are also required for audit purposes (Cilliers, 2017). Electronic health record systems make use of password-based authentication, used in a centralized architecture, which is considered vulnerable to cyberattacks (Mosakheil, 2018). Institutions often make use of insecure clouds to store passwords. These passwords are, more often than not, considered to be weak and can be easily cracked by utilizing techniques such as social engineering, password guessing, and brute-forcing (Kshetri, 2017). Password-based authentication, on its own, is no longer considered secure.

2.2.2 Authorization

Authorization is used to determine the actions an authenticated user can perform on a system (Cilliers, 2017). System functions should be restricted using appropriate authorization mechanisms. Only users that need access to functions should be able to gain access. Restricting rights in this manner mitigate the risk of unauthorized disclosure of information. Electronic health records systems commonly make use of a role-based access control model to enforce authorization (Seol, Kim, Lee, Seo, & Baik, 2018). The role-based access control model provides authorization to users based on the roles assigned to them. A role is generally determined by the user's job function, such as doctors would have the role of doctor assigned to them. Each role is assigned access rights based on a need to know basis (Pussewalage & Oleshchuk, 2016). These roles are generally statically assigned by the system administrator and the role-based model does not support dynamic attributes, for example, control of user permissions based on the time of day. The systems of today need a fine-grained, distributed and dynamic access control mechanism (Bokefode, Ubale, & Modani, 2014).

2.2.3 Audit log

An audit log is a recording of all the actions a user has performed on a system (Dekker & Etalle, 2007). Audit logs are useful in identifying how, when, where, why and by whom data was accessed, modified and/or leaked. Tampering with audit logs frequently occur to cover a criminal's tracks. Healthcare records often get tampered with for various reasons namely, insurance coverage, criminal offenses, and more (Kshetri, 2018). It is

therefore important that electronic health record systems maintain a tamper-proof audit log to identify criminals tampering with healthcare records.

2.2.4 Storing of EHR data

Healthcare institutions traditionally make use of a client-server model to maintain their electronic health records, which are usually stored in a centralized architecture, such as a relational database that represents a single point of potential failure (Liang et al., 2017). Healthcare providers also often make use of potentially insecure clouds to store shared secrets. The U.S. health insurer Anthem's data breach that occurred in December 2014 exposed more than 80 million clients' sensitive information (Kshetri, 2018). This attack is a good example of why centralized architectures are often considered easy and lucrative targets.

2.2.5 Sharing of EHR data

Patients may visit various healthcare institutions throughout their lives. Every healthcare intuitions are required, by law, to record patients' personal and medical information at the first consultation. Patients are often required to provide their medical information from other intuitions, upon request. These processes are time-consuming and could bring about gaps in the patients' medical history. Healthcare institutions are the owners of the medical records they store and as a result are required to comply with regulations, such as the PoPI Act of South Africa, to ensure the confidentiality and integrity of the records (Katurura & Cilliers, 2016). Healthcare institutions are reluctant to share medical records for obvious reasons, for instance, privacy concerns ("A review of cross organizational healthcare data sharing", 2015). This results in patients' medical records being fragmented across multiple institutions (Ekblaw et al., 2016).

When healthcare intuitions do share electronic health records, they make use of three models, namely the push, the pull and the view model. The push model entails healthcare records being sent from one healthcare provider to another. The pull model is utilized when a provider needs to gain access to healthcare records from another provider and the view model enables a provider to view healthcare records, such as x-rays (Catalini, 2017). The United States makes use of a secure email mechanism known as Direct, which enables encrypted communication between healthcare providers. Patients' information can only be pushed between two parties at a time and no other party

can gain access to this transmission. This means that if a patient is transferred from one hospital to another, the latter might not have access to the records held at the initial hospital (Catalini, 2017). The second hospital would need to make a pull request. The traditional models in use lack a standardized means of generating an audit trail (Kshetri, 2017). These models can therefore not ensure data integrity from data creation to data use (Catalini, 2017). Audit trails perform a key role in identifying culprits responsible for data breaches. Although they are technologically sound, data privacy issues are surrounding these models (Kshetri, 2017). Even if institutions can share electronic health records securely, there remains an issue with interoperability as institutions do not all use the same data structures and semantics ("A review of cross organizational healthcare data sharing", 2015).

2.3 South African privacy and data protection regulations

There is currently no specific privacy and data protection statute for electronic health records in South Africa (Townsend, 2017). This study, therefore, focused on the generic privacy and data protection regulations provided by South African law, to ascertain their implications and relevance to electronic health records. Generic privacy and data protection laws discussed in this section are: the South African National Health Act (SANHA); the Health Professions Council of South Africa (HPCSA); the Electronic Communications and Transactions Act (ECTA) and the Protection of Personal Information Act (PoPIA). The summary of each act contained in this review cannot be considered as a legal document but merely serves as a guideline for medical institutions dealing with personal information.

2.3.1 National Health Act No.61 of 2003

The National Health Act provides a framework that strives to realize the rights contained in the Constitution to construct a uniform healthcare system in South Africa (Tuyikeze & Pottas, 2010). The framework outlines the laws that govern local, national and provincial government concerning healthcare services. Chapter 2 of the National Health Act is relevant to electronic health records as Sections 14 to 17 in Chapter 2 outline the provisions related to the confidentiality and privacy of electronic health records (Townsend, 2017). A patient's confidentiality rights are clarified in Section 14 of the National Health Act, which states that all information about a patient is private and

confidential. None of this information is allowed to be shared amongst doctors, workers or any other patients unless the patient personally consents to the sharing of their information. This consent has to be in writing, dated and signed. No person who has access to the patient's health records may disclose any information unless a court order is drawn up, which supersedes the law of confidentiality (South Africa, 2003).

Section 16 of this Act relates to the healthcare provider (Townsend, 2017). Doctors or workers at the institution may only examine the patient's health records for treatment. Further, with the permission of the patient, along with that of the head of the institution and the relevant committees, the information may be used for study or research purposes (South Africa, 2003). This law stands if the identity of the patient is not revealed.

Section 17 relates to the safekeeping of health records (Townsend, 2017). The persons in charge of the health records must set up suitable security measures to prevent unauthorized access to the records. The records must be kept in a safe place with suitable measures. No person may add false information to health records. Health records may only be created, modified or destroyed by authorized persons. No persons may copy any part of a patient's health record without permission to do so. No person may adapt or harm the operating system on which records are kept without permission. Any persons caught breaking this law will be held liable and face a fine, imprisonment or both (South Africa, 2003).

2.3.2 Health Professions Council of South Africa (HPCSA)

The Health Professions Council of South Africa (HPCSA) was formed to enforce the terms of the Health Professions Act (Buys, Chb, Sa, Anaes, & Sa, 2017). The main goal of the HPCSA is to regulate healthcare professionals and to protect patients against maltreatment or abuse inflicted by practitioners employed by healthcare institutions (Townsend, 2017). The HPCSA's regulatory mandate affects both state and privately owned healthcare institutions. Guidelines and ethical rules have been developed by the HPCSA to regulate the ethical importance attached to, for example, the confidentiality and protection of information (Townsend, 2017). These guidelines are presented in the form of booklets.

Booklet 5 (Confidentiality: protecting and providing information)

Booklet 5 outlines the guidelines to be followed by healthcare professionals when handling and protecting the confidentiality of patients' medical and personal information (Townsend, 2017). Section 11 in Booklet 5 presents the guidelines related to the processing of information electronically; doctors and healthcare workers should seek professional advice on how to securely handle confidential information before connecting to any information network. They should make a note of the fact that they took such measures and followed such advice. Doctors, workers, and patients should be confident in knowing that appropriate measures are taken when sending and receiving personal information by any electronic means. Doctors and health care workers must ensure that any information sent by means of fax is encrypted and cannot be seen by anyone other than the intended recipient (Health Professions Council of South Africa, 2016a).

Booklet 9 (Guidelines for the safekeeping of patient records)

Booklet 9 *Section 8 (Alteration of records)*: states that no information may be added, removed or changed in a patient's record without permission to do so. If authorized, changes may only be made by drawing a line through the incorrect information and such change must be dated and signed in full. The reason for the change must be included in the record. The original entry in the record must remain intact and never be removed; the new information may only be added (Health Professions Council of South Africa, 2016b). Section 9 (Duration for the retention of health records) and Section 12 (Retention of patient records on CD-ROM) state that the storage of records on a computer compact disk (CD-ROM) is permitted only if security measures are in place. Only CD-ROMs that allow once only record are to be used so that the information can never be removed. New information can, however, be added. All CD-ROMs and copies thereof must be encrypted and protected with a security password and other measures. The copy of the CD-ROM must be in reading the only format so that no changes can be made to it. The copy must be stored in a different location to the original for safety reasons. The purpose of the copy is so that it can be used to compare the two disks in case of suspicion of tampering. The rights of a patient's confidentiality must always be protected (Health Professions Council of South Africa, 2016b).

Booklet 10 (General ethical guidelines for good practice in telemedicine)

Booklet 10 strives to provide an in-depth guide pertaining to the manner in which healthcare professionals should handle electronic information in a privacy-preserving, confidential and secure manner (Buys et al., 2017). The patient must at all times be assured that their information is private and protected. Patient confidentiality should be ensured at both the consulting and servicing practitioners' sites and should follow the provisions of the Constitution, the National Health Act No 61 of 2003, the Promotion of Access to Information Act No 2 of 2000, the Protection of Personal Information Act No 4 of 2013, the common law and the HPCSA's ethical guidelines pertaining to patient confidentiality in Booklet 10, which generally state that it is every practitioner's obligation to ensure that information is protected effectively at all times (Health Professions Council of South Africa, 2014).

HPCSA's booklet on confidentiality further provides guidelines on the manner and conditions under which patient information may be disclosed, for example in the case of research, education, clinical audit, financial auditor or for the publication of case histories and photographs, if permitted to do so. Policies and procedures for the documentation, maintenance, and transmission of records regarding telemedicine consultations should be held to the same standard of care as face-to-face consultations (Health Professions Council of South Africa, 2014). Policies and procedures pertaining to telemedicine should deal with the confidentiality, healthcare personnel, (apart from the healthcare practitioners), who will process the electronic information, the time spent, types of electronic transactions permitted and the necessary patient information that has to be included in electronic communications.

It is the responsibility of the healthcare personnel to adhere to safety measures when working with personal information (Health Professions Council of South Africa, 2014). Prescriptions, test results or any other information sent by electronic means must be safely secured with passwords, encryption and/or any other reliable measures. Healthcare practitioners utilizing telemedicine should avoid accidental damage and loss of patient information and provide safe procedures to avoid any adjustments or removal of patient data. They must ensure that patient information obtained electronically is kept in line with the HPCSA's advice with regard to keeping patients' records safe in Booklet 15 and comply with the legal requirements for data messages contained in the Electronic Communications and Transactions Act No 25 of 2002 regarding the protection of

information and the principles regarding the electronic collection of personal information (Townsend, 2017).

2.3.3 Electronic Communications and Transactions Act 25 of 2002 (ECT Act)

The Electronic Communications and Transactions (ECT) Act outlines the approved data protection rules relevant to personal information (Townsend, 2017). Personal information is defined in the ECT Act as any information relating to a person's private life that is private and confidential, be it information pertaining to age, sex, race, gender, nationality, origin, health, religion, culture, pregnancy, marital status, education, employment history, or finances. The person's address, blood type, and fingerprints, as well as any identifying number or symbol allocated to the individual. An individual's preferences, views, and opinions are personal. Information about a person's views concerning another person may not be shared. Any information about a person is confidential, except material relating to a person who has been deceased for more than 20 years (South Africa, 2002).

According to Section 51, data handlers should first obtain consent from the data subjects themselves before any personal information can be collected, processed or disclosed (Townsend, 2017). Data handlers are only permitted to use personal information for the reason it was collected, unless consent has been granted by the data subject or by a court order (South Africa, 2002). Disclosure of personal information to third parties is also prohibited unless consent has been provided by either the data subject or a court order. Personal information that has become obsolete should be deleted and destroyed as soon as possible. Data handlers are permitted to make use of personal information for statistical purposes only after removing all identifying information (South Africa, 2002). Only the statistical results may be disclosed freely. The ECT Act does not delve too deeply into the security requirements related to handling personal information. The PoPI Act is geared towards the protection of personal information and will most likely replace the ECT Act in the future concerning the processing of personal information (Townsend, 2017).

2.3.4 Protection of Personal Information ACT 4 of 2013 (PoPI)

In 2013, the then President of South Africa, Jacob Zuma, signed the protection of personal information (PoPI). The law came into effect on 26 November 2013. The main

objective of the PoPI Act is to advocate and safeguard personal information, handled by the private and public domain. The PoPI Act provides substance and effect to the privacy rights contained within the Constitution (Townsend, 2017). The PoPI Act strives to mirror international privacy standards (Swartz, 2017).

Personal information processing stipulations

The PoPI Act defines personal information as any information relating to a person's private life; information pertaining to age, sex, race, gender, nationality, origin, health, religion, culture, pregnancy, marital status, education, employment history and finances (South Africa, 2013). The person's address, blood type and fingerprints, as well as any identifying number or symbol allocated to an individual. The processing of personal information should be conducted lawfully to protect patients' privacy.

The PoPI Act outlines eight principles for safeguarding the lawful processing of personal information. The conditions for the lawful processing of personal information are categorized as follows: accountability; processing limitation; further processing limitation; purpose specification; information quality; openness; security and data subject participation (South Africa, 2013). The data subject should first be made aware of why and how information will be collected and processed. The purpose of the processing of personal information should always be clearly explained to the data subject. A data subject is any person who can be uniquely identified by criteria such as a name or an identity number etc. Data subjects should provide consent before any personal information is collected or processed in any way or form. The data subject should be of sound mind when providing consent to access their personal information. Data should only be collected from the data subjects themselves and should not be obtained from a third party (South Africa, 2013). Only the relevant personal information should be collected to minimize the privacy exposure of a data subject, for example photographs should contain only the patient's injury and not the full body unless relevant. Data handlers are only permitted to use personal information for the reason for which it was collected, unless consent has been given by the data subject or by a court order (South Africa, 2013).

The data handlers should always ensure that the personal information collected is accurate and complete. Personal information should also be updated as frequently as possible. Personal information that has been collected should be stored only as long as

necessary. The minimum retention period for medical information is five years. This may be extended for historical statistical and research purposes. The information should be destroyed/deleted/all identifying markers removed as soon as reasonably possible (Buys et al., 2017). If the data is breached during the retention period the breach is indefensible and the data handlers can face hefty fines. Access to, and processing of, personal information should be restricted to authorized personnel. The data subject should always be made aware of possible breaches to their privacy. Only the court and cabinet can authorize the processing of personal information without the consent of the patient. This infringement on patients' privacy should only occur when the public or private interests supersede the right to privacy. In cases such as national security and criminal prosecution and investigation.

Section 19 states that a data handler is responsible for the integrity and confidentiality of the personal information they store (South Africa, 2013). Data handlers should identify all present or future risks to personal information from internal and external threats. These identified risks should be mitigated through the implementation of commonly accepted information security controls (Townsend, 2017). Employees under the authority of a data controller are only permitted to disclose personal information if it is required for performing their professional duties (South Africa, 2013). This is relevant to the hospital staff working under the authority of a healthcare institution.

Authorization of sensitive personal information

Sensitive personal information includes the following: the religious or philosophical beliefs, race or ethnic origin, trade union membership, political persuasion, health or sex life or biometric information of a data subject is considered as sensitive personal information (South Africa, 2013). Section 26 state that data handlers are prohibited from processing sensitive personal information. However, the prohibitions do not apply to certain intuitions such as social services and healthcare institutions (Townsend, 2017).

Table 2.1: Comparison of South African policies.

Criteria	**SANHA**	**HPCSA**	**PoPI**
Authorization	Intuitions are required to set up suitable security masseurs to prevent unauthorized access to health records.	All computers used to store or process electronic health records in any form should only be accessed by authorised personnel through the use of a login password. No unauthorized person should be able gain access to health records.	Access to and processing of personal information should be restricted for authorized use only.
Storage	Health records should be stored in a safe place with suitable security measures.	Health records in electronic format should be safeguarded with security measures, e.g., encryption. The use of ROM technology, e.g., CD-ROM is permitted. Provided that copies are made and stored in a different physical location for safety reasons.	Data handlers should identify all present or future risks to personal information from internal and external threats. These identified risks should be mitigated through the implementation of commonly accepted information security controls.
Sharing	Sharing of health records with any party is strictly prohibited unless the patient provides consent. Only a court order can trump this prohibition.	Any personal information shared electronically should be safeguarded with security measures, e.g., passwords, encryption, and/or any other reliable security.	Only a court order can authorize the sharing of personal information without the consent of the data subject. Data handlers are allowed to use de-identified personal information for statistical purposes. Only the statistical results may be disclosed freely.

Table 2.1: Comparison of South African policies. (continued)

Criteria	**SANHA**	**HPCSA**	**PoPI**
Immutability	No health records may be created, modified or destroyed without authority to do so.	The original entry of a health record must stay intact and never be removed. New or modified information should only be appended to the health record.	Personal information collected should only be stored for as long as necessary. Personal information should be destroyed/deleted/de-identified as soon as reasonably possible. The data subject is permitted to request for their personal information to be destroyed.
Audit logs	When patients provide consent it needs to be date and signed by the patient.	Changes made to health records should be signed and dated by the person making the changes. The reason for the change should also be stated.	

2.4 Comparison

This section serves as a comparison of the policies and regulations with regard to electronic health records in South Africa. The PoPI act overlaps and supersedes the Electronic Communications and Transactions (ECT) Act (Townsend, 2017). Therefore, the ECT Act has been omitted from this comparison. Table 2.1 contains a comparison of the following policies: South African National Health Act (SANHA); Health Processions Council of South Africa (HPCSA); Protection of Personal Information Act (PoPI).f

2.5 Conclusion

Chapter 2 outlined various challenges and privacy policies related to the safeguarding of patients' personal and medical information thereby addressing secondary objective one. The common theme among these policies is that the healthcare intuitions are responsible for the integrity and confidentiality of a patient's personal and medical information. Patients are required to provide consent to an institution before any personal or medical information is collected or processed in any manner. The challenges currently faced by electronic health record systems pose a significant risk to the privacy and integrity of patients' health records.

Chapter 3

Distributed ledger technology : Blockchain technology

3.1 Introduction

Blockchain technology is a type of distributed ledger technology that differs from other distributed ledger technology by bundling unrelated data into blocks that are chained together in a linked-list manner, hence the name blockchain (Bashir, 2017, p. 18). The blockchain innovation is fundamentally built on old technologies used in new ways. Blockchain technology is based on cryptography that predates back to the 1980's. Incremental advances and new ways of thinking resulted in the novel blockchain protocol as we know it today. David Chaun's dissertation "Computer Systems Established, Maintained, and Trusted by Mutually Suspicious Groups" that was introduced in 1982 outlined various aspects of a blockchain-like distributed system that lack some key features present in blockchain protocols today (Sherman, Javani, Zhang, Golaszewski, & County, 2019). The first full-fledged blockchain protocol known as Bitcoin was introduced in 2008. Today Bitcoin serves as a landmark and starting point for all things blockchain. The terms Bitcoin and blockchain are being used interchangeably. These two terms, however, are not the same. Blockchain technology is the underlying technology used in the Bitcoin protocol to facilitate the secure transfer of Bitcoin (Bashir, 2017, p. 111). The term 'Bitcoin' is the name of the cryptocurrency that powers the Bitcoin network. Blockchain technology is not limited to the application of cryptocurrency (Bashir, 2017, p. 23).

Blockchain, in simple terms, can be seen as a distributed database controlled by a group of individuals. When a user wishes to add a record to the database in blockchain terms they purpose a transaction. This transaction is then broadcast to a peer-to-peer network consisting of computers known as nodes. The network of nodes validates the transaction and the user's status using known algorithms. A verified transaction can involve cryptocurrency, records and smart contracts (Bashir, 2017, p. 42). Once the transaction has been verified, it is combined with other transactions to form a block of data. This block of data is then added to the blockchain in a way that is permanent and

unalterable (Bashir, 2017, p. 27). This chapter serves to address secondary objective two which aims to determine which aspects of blockchain technologies are most suitable for the secure storage and distribution of information.

3.2 Cryptography

This section will outline some of the cryptographic components that form part of a blockchain data structure.

3.2.1 Cryptography Hash Function

A cryptographic hash function is a one-way mathematical function/algorithm that takes any form of data and generates a unique string of characters known as a hash. Hashing functions can take any type, length or size of data. The result would always be a unique string of identical lengths. The difference between hashing and encryption is that encryption can be reversed, or decrypted by using a special key. Hashes, however, can not be deciphered or reversed by ruining the mathematical function in reverse (Thakur, 2017). Therefore, it is very difficult to reverse a cryptographic hash. The only way to uncover data that has been hashed is to guess what the data is and run it through the hashing function until the hash string matches the target one. When hashing identical data, the output will always result in the same hash. The most popular hashing functions are MD5, SHA1, and SHA-256. Blockchain technology currently makes use of SHA-256 (Kedar Iyer, Rene Madsen, Solomon Lederer, Michael Wuehler, Joseph J. Bambara, 2018, P. 20). Hashing with SHA-256 would result in a unique string of 64 characters. Refer to Figure 3.1 for an example of a SHA-256 hashing function in action. Hasing is also used to build a Merkle tree as described in Subsection 3.2.2.

3.2.2 Merkle Tree

A binary tree of hash pointers also known as a Merkle tree is efficient at verifying the integrity of sizeable data structures. Merkle trees form an elementary part of blockchain networks (Bashir, 2017, p .95). Blockchain networks make use of Merkle trees to securely verify blocks as they are distributed across a peer-to-peer network. Merkle trees often contain a sizable amount of blocks that usually contain numerous transactions or data. A Merkle tree is build by continuously hashing pairs of nodes until only a single node remains, known as the Root hash or Merkle Root. This process starts with the leaf-nodes, at the bottom, and is built up to the Merkle node. Each leaf node contains a hash of transactional data. Normal nodes contain a hash of the combined hashes of the previous two nodes (Thakur, 2017). Merkle trees are binary and as a result, require a pair of two nodes to built up to the root hash or Merkle root. If the number of leaf-nodes is odd the last hash would be duplicated to even out the tree. When changes are made to the data stored in leaf-nodes the process of rebuilding the tree would result in a different root hash compared to the original root hash. Therefore, Merkle trees can be

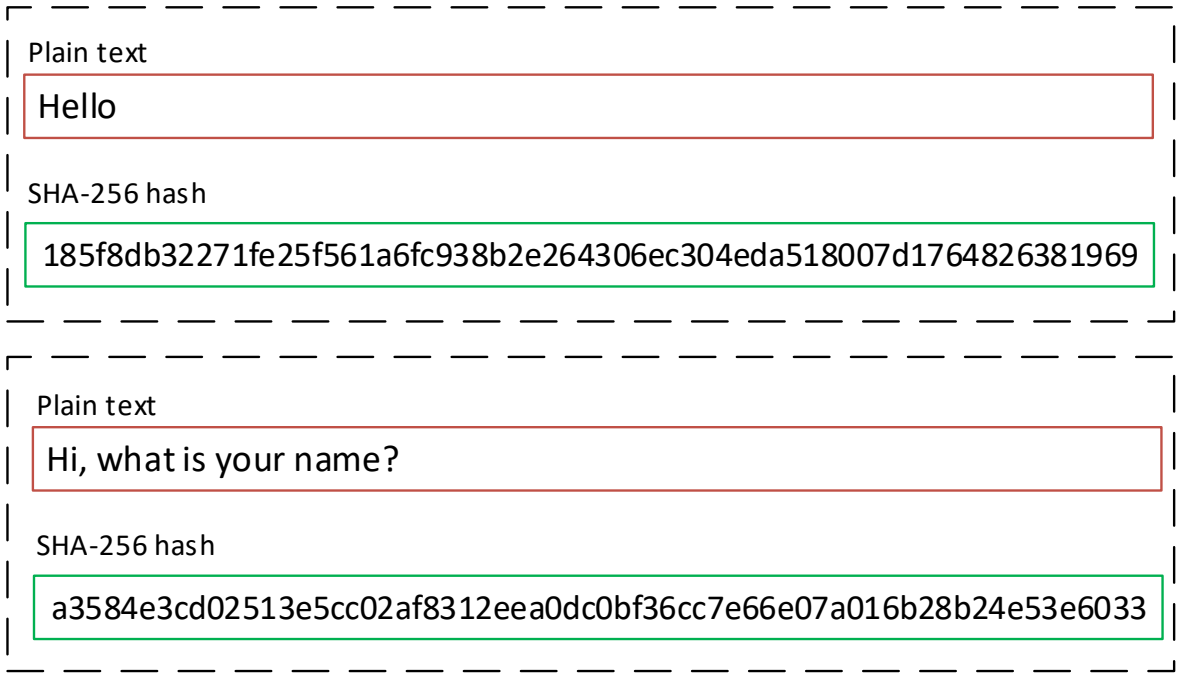

Figure 3.1: SHA-256 hash

used to verify the integrity of data by comparing the root hashes. In a blockchain, each block contains a Merkle root of all the transnational data stored in a block. The block is then hashed with all of the contents of the block including the previous block's hash and Merkle root hash (Kedar Iyer, Rene Madsen, Solomon Lederer, Michael Wuehler, Joseph J. Bambara, 2018). Refer to Figure 3.2 for an example of what a blockchain looks like in terms of the Merkle root.

3.3 Blockchain technology components and features

Blockchain technology consists of numerous components and features with desirable properties. These properties are as follows: availability: the shared ledger mitigates the risk of a single point of failure; decentralization: allows parties that do not trust one another to share information without relying on a central authority; transparency: the shared ledger promotes transparency between parties; integrity: is preserved through the use of an immutable shared ledger; The consensus mechanism is used to prevent or detect any tampering with the shared ledger (Emmadi, Vigneswaran, Kanchanapalli, Maddali, & Narumanchi, 2019). This section will further outline the components and features concerning various aspects of blockchain technology.

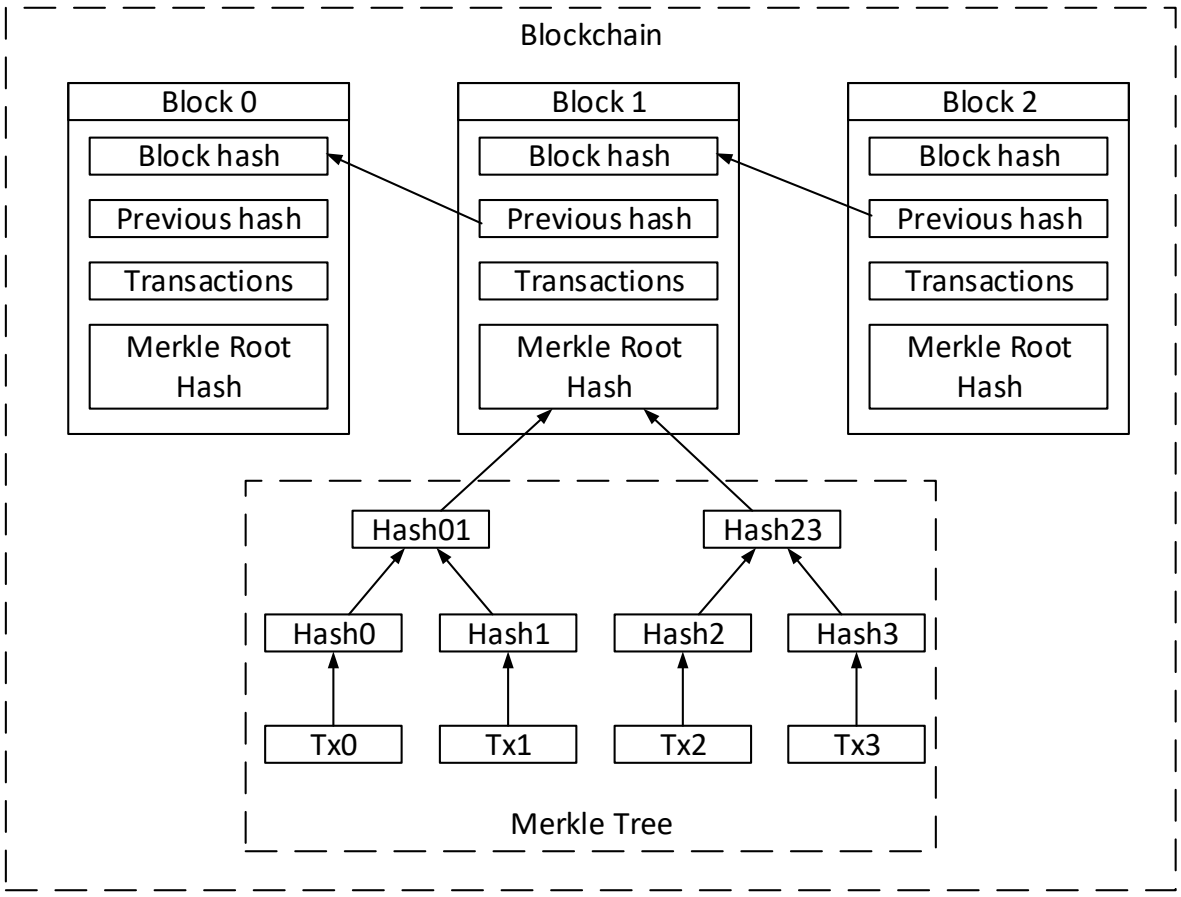

Figure 3.2: Merkle tree and blockchain

3.3.1 Types of blockchain technology

Blockchain technology has been evolving over the years and it can now be divided into various types with distinct attributes (Bashir, 2017, p. 25).

3.3.1.1 Public

A public blockchain is a large distributed network that runs with the support of a native token. This type of blockchain network is open to participation by anyone at any level and transactions in such a network are visible to all participants (Bashir, 2017, p. 26). It is difficult to establish and control the identity of participants (Corporation, 2017). The source code of this type of protocol is generally open-source, for example, Bitcoin.

3.3.1.2 Permissioned

A permissioned blockchain is normally a large, distributed network that runs with or without the support of a native token. Permissioned blockchains are mostly run by a consortium of organizations that need to share information in a trusted manner (Cachin

& Vukolić, 2017). The roles that an individual can perform in a permissioned network are controlled and transactions in such a network are generally confidential between participants (Corporation, 2017). Permissioned blockchain networks hit a sweet spot between public and private blockchain networks.

3.3.1.3 Private

A private blockchain is normally a small, distributed network that does not make use of a native token. Private blockchains are mostly owned and controlled by a single organization (Cachin & Vukolić, 2017), the membership of which is strictly controlled. Transactions in such a network are secret and all the participants' identities are known (Corporation, 2017).

3.3.2 Consensus algorithms in blockchain technology

In basic terms, consensus could be illustrated as a group of individuals mostly agreeing on the same decision. For example, the consensus among a group of travelers might be to go swimming on a hot day. The minority of the individuals might not agree but the consensus is to go swimming. The concept of consensus is used by blockchain technology to ensure that all the peers in a network agree to a single history of transactions. (Bergquist, 2017). There are mainly two types of consensus algorithms, namely proof-based and Byzantine fault tolerance based. The proof-based consensus is where a leader is elected based on having some kind of proof that grants them the authority to propose a new value. Byzantine fault tolerance based consensus is where rounds of votes are used to propose a new value (Bashir, 2017, p. 28). Examples of consensus algorithms include proof-of-work, proof-of-stake, proof-of-elapsed-time, proof-of-authority, practical Byzantine fault tolerance and more.

3.3.2.1 Proof-of-work (PoW)

The proof-of-work algorithm is used to secure the transaction history of a public blockchain network against tampering (Baliga, 2017). Each node in the network competes to solve a computational resource-intensive puzzle. The node that solves the puzzle first is rewarded with cryptocurrency and is authorized to append a new block of data to the blockchain network. These nodes that compete to solve the puzzle are known as miners (Ekblaw et al., 2016), which are highly specialized types of computers that are optimized to solve these computational puzzles. Miners use an excessive amount of electricity and are expensive to purchase (Bashir, 2017, p. 133). The proof-of-work algorithm is more suited to public blockchain networks (Anh, Zhang, Ooi, & Chen, 2018).

3.3.2.2 Proof-of-Stake (PoS)

The Proof-of-Stake algorithm works on the notion that if a user has invested enough in the system they will not benefit from attacking the system (Baliga, 2017). Proof of stake does not make use of mining nodes but of validating nodes, which are chosen in

a deterministic fashion based on a security deposit referred to as a stake. The node with the highest stake has the highest probability of being chosen as the validating node of a new block (Siim, 2017). The validating node receives a reward for each block validated. If a validating node approves a fraudulent block it is penalized and a portion of the deposit(stake) is removed. This algorithm keeps the network secure by punishing fraudulent behavior. Users will not attack the network if the punishment outweighs the reward for executing an attack. The proof-of-stake algorithm is more suited to public blockchain networks (Anh et al., 2018).

3.3.2.3 Proof-of-elapsed-time (PoET)

The Proof-of-elapsed-time algorithm randomly selects a leader node after a set period (Baliga, 2017). The leader node can propose a new block. The timers used in the system run in an Intel SGX supported CPU. The timer value of each node is set by a probability distribution F, which is predetermined by the network. Intel SGX ensures that each node's timer can be trusted. When a node's timer has timed out, the SGX generates a quote that can easily be verified by other nodes in the network. The quote represents the leader's proof that it has waited long enough before adding a new block. This quote can be statistically verified by each node in the network to ensure its validity. Intel SGX is not considered to be 100% secure (Chen et al., 2017). Side-channel attacks pose a real threat to Intel SGX (Lindell, 2018). According to Intel, it is the developer's responsibility to implement countermeasures to mitigate such attacks.

3.3.2.4 Proof-of-Authority (PoA)

The Proof-of-Authority algorithm averts a single point of failure by scheduling work between trusted nodes known as authority nodes, which are predetermined. Only authority nodes can add new blocks to the blockchain network. The Proof-of-Authority algorithm makes use of round-robin scheduling to provide each authority node with a fair amount of time to propose new blocks (Anh et al., 2018).

3.3.2.5 Practical Byzantine fault tolerance (PBFT)

Practical Byzantine fault tolerance (PBFT) was first introduced in 1999 by Barbra Liskov and Miguel Castro. This algorithm enables state machine replication that can effectively mitigate against Byzantine nodes (Bashir, 2017, p. 30). This means that PBFT can detect malicious nodes. The PBFT algorithm requires 3f+1 number of nodes to work effectively. Rounds of votes are issued to achieve consensus and detect faulty nodes. The PBFT algorithm does, however, come with scalability issues. The number of messages increases exponentially with each node added to the network. Documented experiments have only scaled up to twenty nodes successfully (Baliga, 2017).

3.3.2.6 Tendermint

Tendermint has a high throughput with an established amount of validators and can easily track malicious nodes (Tendermint, 2018). The Tendermint consensus algorithm works in the following way a transaction is proposed and distributed to all validators. The proposal needs to be received by all of the validators in a set amount of time otherwise the transaction proposal is discarded (Saraf & Sabadra, 2018). Once all the validators have received the transaction they start a two-phase voting process known as a pre-vote and pre-commit (Tendermint, 2018). If two-thirds of the validators voted for a transaction in both voting phases it can then be appended to the blockchain. Tendermint also makes use of locks to ensure that validators can't commit different blocks with the same height associated (Saraf & Sabadra, 2018). Each block has a height value. The highest value is considered to be the latest block to be appended to the blockchain (Tendermint, 2018). Locks ensure that the blockchain does not split into two chains (Saraf & Sabadra, 2018). One of the disadvantages of this consensus mechanism is that the system may halt if a third of the validators are offline (Saraf & Sabadra, 2018).

3.3.2.7 Raft

Raft is a consensus algorithm that controls a distributed log between trusted peer nodes. Raft is normally implemented in a clustered formation comprising an odd number of nodes. Clusters typically consist of five nodes, which ensures that the raft protocol is fault-tolerant. Note that the Raft consensus algorithm is fault-tolerant and not Byzantine fault-tolerant. Raft cannot detect malicious nodes in the network. Raft nodes can be in one of three states, namely follower, candidate or leader. The leader node is responsible for attending to client requests and replicating the state machine logs between nodes. Followers can only respond to requests from leader and candidate nodes. Client requests sent to a follower node are automatically forwarded to the leader node. Candidate nodes are nodes that are next in line to assume a leader's duties in the event of a leader failing. The raft protocol first elects a dedicated leader with complete authority to manage and replicate the log and respond to log entry requests from clients. The log entries are then validated by the leader node and distributed across the network. Nodes vote for a new candidate in the event of a leader failing or reaching the end of its term. The candidate node with the highest number of votes is then promoted to the leader node. Leaders can only lead for a set period known as a term (Leibovici, 2015).

3.3.2.8 Apache Kafka and Zookeeper

Kafka is a distributed messaging system first developed by Linkedin (Hiraman, 2018). The Kafka system is open-sourced under the Apache license. Kafka collects messages in the form of bytes and then appends them to a queue represented as an array. Messages are published to a specific topic by a Producer. Each topic contains a single isolated message queue. Kafka can handle thousands of topics. A Topic can be replicated across multiple Kafka nodes in the form of a partition. Partitions enable parallel reads on a message of a specific topic. Applications that read messages from topics are called

Consumers. Consumers use offset values to keep track of their read position in each partition. Kafka nodes are clustered together with an odd number of nodes to ensure fault tolerance. Thus Kafka cluster configurations typically consist of 3, 5 or 7 nodes. Zookeeper is used to manage consumers and the nodes in the Kafka cluster. Consumer's offset values are stored and managed within Zookeeper as Kafka nodes are stateless. Zookeeper manages which Kafka node is a leader of which topic partition. This metadata is forwarded to the producers and consumers on the network. It should be noted that Kafka cannot properly function without Zookeeper (Kumar & Singh, 2017). Apache Kafka and Zookeeper are used together in Hyperledger Fabric to keep ordering nodes in sync (Bashir, 2017, p. 362). Hyperledger fabric uses Kafka and Zookeeper as they were designed to be fast reliable and scalable.

3.3.3 Smart contract

Smart contracts were proposed in the late 1990s by Nick Szabo (Bashir, 2017, p. 198). A smart contract in blockchain terms is used to digitally facilitate, enforce and verify that all the terms, of a contract (business logic), are met before a transaction can take place (Bergquist, 2017). Smart contracts eliminate the need for third parties involvement. Once a smart contract is executed it is tractable and irreversible (Bashir, 2017, p. 198). Thus the logic of a smart contract should be thoroughly tested before it is deployed on a production blockchain network.

3.4 Is blockchain technology the right fit?

Blockchain technology is not suited for every use case and it can be difficult to select the appropriate blockchain technology type. The flowchart in Figure 3.3 could help in selecting the correct blockchain technology to use. This flowchart was derived from (Emmadi et al., 2019). Enterprise blockchain networks, also known as permissioned blockchains, are being developed to cater for use-cases of enterprise use. Public blockchain networks lack key features such as confidentiality, privacy, user identity, authorization and audibility, and these features are thus being incorporated into permissioned blockchain networks to accommodate enterprise use cases (Emmadi et al., 2019).

3.4.1 Privacy and confidentiality

One of the challenges for the realization of enterprise-grade blockchain technology is privacy (Bashir, 2017, 461). Enterprise entities that are part of a consortium network require information to be shared privately and securely. Consortium networks also require a level of isolation to ensure that only authorized parties can gain access to confidential information. The nature of blockchain technology is to promote transparency through the use of a shared ledger. Transparency and privacy are considered to be opposing forces and permissioned blockchain technology, therefore, strives to achieve a balance between transparency and privacy (Emmadi et al., 2019). Research is currently underway to

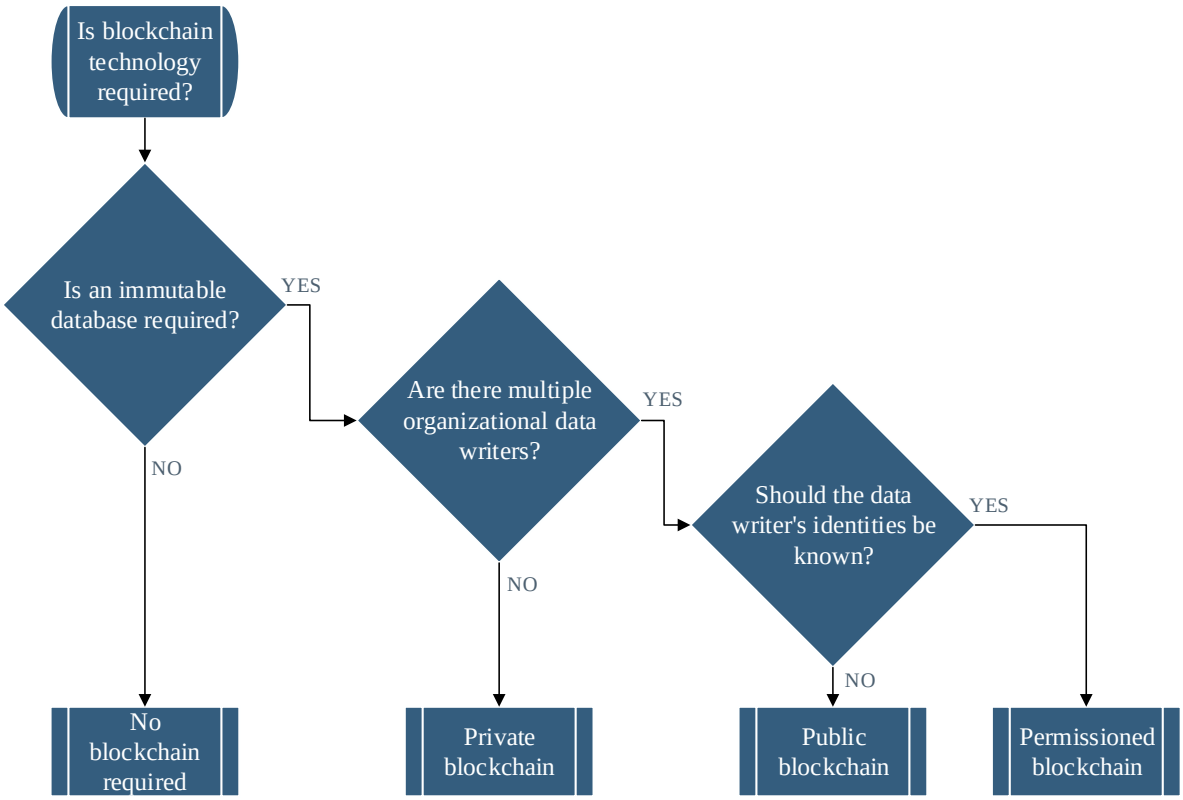

Figure 3.3: Blockchain type flowchart

improve the level of privacy and confidentiality offered by blockchain technology. This section covers identified cryptographic methods for enhancing the privacy and security of blockchain technology.

3.4.1.1 Indistinguishability obfuscation

Indistinguishability obfuscation was first proposed by Shai and others in their research paper *Candidate Indistinguishability Obfuscation and Functional Encryption for All Circuits* in 2013. The indistinguishability obfuscation technique is used to convert readable program code into unintelligible program code whilst preserving the program's functionality (Garg et al., 2013). Blockchain technology could utilize an indistinguishability obfuscation mechanism to convert smart contracts into a black box system (Bashir, 2017, p. 450).

3.4.1.2 Zero-knowledge proofs (ZKP)

Zero-knowledge proofs were introduced by Rackoff, Goldwasser and Micali (Bashir, 2017, p. 451) to enable a prover to demonstrate to a verifier that they have knowledge about a secret without exposing too much information about the secret (Bashir, 2017, p. 451). Zero-knowledge proofs consist of three components, namely completeness, soundness

and a zero-knowledge component. Completeness guarantees that if a statement about a secret is true, the verifier will be convinced by the prover's knowledge about a secret. Soundness guarantees that if a statement about a secret is false, the verifier should not be able to be persuaded by a deceitful prover. The zero-knowledge component ensures that nothing about the secret is revealed when proving the existence of that secret. Zero-knowledge proofs, in blockchain terms, could ensure that transactions are validated without revealing information about the recipient, the sender and the actual transaction details (Bünz et al., 2017).

3.4.1.3 Homomorphic encryption

Homomorphic encryption is a method of encryption that enables the use of arithmetic operations on encrypted data, thus producing an encrypted result (Bashir, 2017, p. 451). Performing the same arithmetic operation on the plaintext would match the result of the encrypted value when decrypted (Song & Wang, 2017). Until recently, only partial homomorphic schemes existed that could either be additive or multiplicative. Fully homomorphic encryption was discovered by Craig Gentry in 2009 and this encryption scheme supports both additive and multiplicative operations (Song & Wang, 2017). Fully homomorphic encryption is currently not efficient enough to be used in production. Blockchain technology could utilize full homophobic encryption to increase the privacy and confidentiality of transactions. Homomorphic encryption could be applied to data before it enters the blockchain network, thus enabling arithmetic operation without having to decrypt the data. This would ensure privacy on the blockchain network (Bashir, 2017, p. 451).

3.4.1.4 State channels

A state channel is a communication pipeline (channel) between users and a service (Coleman, Horne, & L4, 2018). The users of a channel create an agreement and compliance is then enforced by coding the agreement into a smart contract. Messages sent in a channel are referred to as transactions. Each transaction is signed by the sender to ensure the audibility of the transactions later (Coleman et al., 2018). These transactions are run off-chain, which means that the blockchain network has no knowledge of any of the transactions (Bashir, 2017, p. 451). The final result of the transactions is posted as a single transaction on the blockchain network.

3.4.1.5 Intel Software guard extension (SGX)

Intel Software Guard Extension (SGX) is an instruction set embedded in the Intel processors that enable code execution in a hardware protected runtime environment known as an enclave (Brasser et al., 2017, p. 452). Enclaves are isolated from other software running on the system, for example, the BIOS, the operating system (OS) and the hypervisor (Brasser et al., 2017). It should be noted that enclaves are isolated from one another. SGX can be used in blockchain technology to secure the execution of smart

contract code and enable the secure use of the proof-of-elapsed-time consensus algorithm (Bashir, 2017, p. 452).

3.4.2 Authentication and identification

Enterprise applications need to be able to identify the users interacting with the system for security and audit purposes. Permissioned blockchain networks currently make use of public key infrastructure (PKI) to authenticate and identify users in the blockchain network (Emmadi et al., 2019). This type of authentication is known as certificate-based authentication. The administrator of a network would register users with a certificate authority (CA). The CA is responsible for identifying, creating and revoking digital certificates (Hyperledger, 2019, p. 49). A user can enroll with a CA by issuing a certificate signing request (CSR) to retrieve their digital certificate. Digital certificates generally contain the user's identity and public key (Hyperledger, 2019, p. 47). Digital certificates can also be used to identify devices connected to the network. When a user or device would like to interact with the blockchain network it needs to attach its certificate to the transaction. The peer node receiving the transaction validates the digital certificate by sending it to the CA. The CA validates the signature of the certificate and compares it to the certificate revocation list (CRL), which contains a list of invalid or expired certificates (Hyperledger, 2019, p. 51). This allows the revocation of access and enables the setting of an expiration date on the digital certificate. This process is considered more secure than the traditional username and password authentication. Digital certificates should be stored securely with the use of security measures, e.g. storing certificates on a hardware security module (HSM).

3.4.3 Authorization

Current enterprise systems require a fine-grained access control model, as the traditional role-based access control (RBAC) model is no longer considered adequate. Attribute-based access control (ABAC) has been proposed to provide a more fine-grained access control mechanism for enterprise systems (Emmadi et al., 2019). The ABAC and RBAC models both rely on policy-driven implementation. Attribute-based access control differs from RBAC in that the model combines attributes of users and objects to make access control decisions instead of utilizing user roles. The RBAC model requires the user to be registered to access the system. The ABAC model does not require users to be registered to be able to interact with the system. ABAC is therefore suited for systems that require a more dynamic and fine-grained access control mechanism, for example, EHR systems (Pussewalage & Oleshchuk, 2016). Attribute-based access control could be used to restrict access based on attributes such as user roles, user location, the time of day and more. This effectively means that a user with a certain role and location can only access information at a specific time. Permissioned blockchain technology makes use of the attribute-based access control (ABAC) model. The access control rules are embedded in the smart contracts code (Emmadi et al., 2019). If the access control rules need to be changed, the smart contracts need to be modified and redistributed to

the blockchain network. Permissioned blockchain technology provides a distributed and tamper-evident access control mechanism.

3.4.4 Audibility

Public blockchain networks cannot be regulated and as such cannot be audited because the identities of participants are pseudo-anonymous. Enterprise applications are audited regularly and an auditor should, therefore, be able to access and identify users on the blockchain network. The level of access granted to auditors may vary based on the requirements (Emmadi et al., 2019). Permissioned blockchain networks make use of membership service to identify users interacting on the network and permissioned blockchain technology could thus be used to record all activities on a network resulting in a standardized immutable audit log (Bashir, 2017, p. 418). Isolation could be used to restrict the level of information available to the auditor to preserve users' privacy.

3.5 Popular blockchain protocols

This section outlines popular blockchain protocols and their features. The summary of each blockchain protocol was accurate at the time that this study was undertaken. The information provided in this section could no longer be accurate, as the blockchain protocols evolve rapidly.

3.5.1 Bitcoin

Type: Public
Consensus: Proof-of-work
Privacy: n/a

Bitcoin was introduced in a White Paper in the autumn of 2008. The Bitcoin open source software was released in 2009 and the founder of Bitcoin remains anonymously known as Satoshi Nakamoto (Laurence, 2017, p. 32). Bitcoin is a popular cryptocurrency, the success of which sparked the blockchain revolution. Bitcoin makes use of an extensive consensus algorithm known as proof-of-work to validate transactions. Proof-of-work is known by the Bitcoin community as "mining". Bitcoin miners use highly specialized equipment that is not only expensive but also consumes large amounts of electricity to operate. Mining is necessary to keep the Bitcoin network safe, stable and secure (Laurence, 2017, p. 34).

3.5.2 Hyperledger

The Hyperledger project was initiated in 2015 by the Linux Foundation and is led by Executive Director, Brian Behlendorf (Laurence, 2017, p. 81). Brian has decades of experience with the Linux and Apache Foundations and is a CTO at the World Economic Forum. This project has formed numerous partnerships with large organizations such

as IBM. There are currently five frameworks under the Hyperledger umbrella, namely Fabric, Iroha, Sawtooth Lake, Indy and Burrow (Hyperledger Architecture Working Group, 2017). The Hyperledger's first production-ready project/framework was released in July 2017 and is known as Hyperledger Fabric.

3.5.2.1 Hyperledger Fabric

Type: Permissioned
Consensus: Kafka
Privacy: Zero-knowledge proofs, Channels

Hyperledger Fabric is a permissioned blockchain platform, contributed by IBM and Digital Asset. The Fabric project strives to address issues such as scalability, privacy, and confidentiality (Bashir, 2017, p. 362). Fabric serves as a foundation layer to deliver blockchain networks that are suited for business. With Hyperledger Fabric many of the modules are pluggable. For example, the developers can select a consensus algorithm that is appropriate and plug that into the network (Saraf & Sabadra, 2018). This provides a high degree of flexibility and scalability that most blockchain applications lack, for example, Bitcoin. Hyperledger Fabric enables the creation of channels that make use of peer-to-peer technology (Bashir, 2017, p. 362). This enables participants to share confidential information. This information can only be viewed by participants on that particular channel. Participants can belong to many channels on the same network. Fabric makes use of container technologies that can host any programming language. This enables developers to program chain-code commonly known as smart contracts in various languages, for example, Go, Java, and Node.js (Bashir, 2017, p. 362). A transaction in a Fabric network is private, confidential and anonymous to normal users. This transaction can, however, be traced and linked to users by authorized auditors. Fabric being a permissioned blockchain network means all participants need to be registered with the membership service in order to gain access to the network (Saraf & Sabadra, 2018).

3.5.2.2 Hyperledger Burrow

Type: Permissioned
Consensus: Tendermint
Privacy: n/a

The Burrow platform originated from a project known as Monax and is currently being developed by the Linux Foundation under the Hyperledger project umbrella. Burrow's main objective is to add permissioned blockchain support to the Ethereum blockchain and the Burrows protocol, therefore, supports smart contracts written in Solidity. Ethereum is a popular permissionless blockchain platform. Adding permissioned blockchain support to the Ethereum blockchain makes it more flexible and attractive to business use cases. Hyperledger Burrows consist of three main components referred to as the consensus engine, the Ethereum Virtual Machine (EVM) for smart contract

support and the remote procedure call (RPC) gateway. Burrows could thus be seen as an Ethereum client adding permissioned support to the existing Ethereum network. The consensus mechanism used by Burrows is known as Tendermint, which has a high throughput with an established amount of validators and can easily track malicious nodes (Saraf & Sabadra, 2018). Hyperledger Burrows is still in the incubation phase and is expected to be released in 2019.

3.5.2.3 Hyperledger Iroha

Type: Permissioned
Consensus: Sumeragi-BFT
Privacy: Channels

Hyperledger Iroha was first introduced in September 2016 by Soramitsu, Hitachi, Colu and NTT Data (Bashir, 2017, p. 356). The Iroha project's main focus is to complement other Hyperledger projects by creating metamorphic components written in C++ geared towards mobile development. The Iroha platform includes a built-in function that makes it easy to execute simple tasks. Custom smart contracts are also supported and can be written in Java. Iroha differs from other blockchain platforms in the way that it exerts control over who is allowed to store which segments of the data history (Saraf & Sabadra, 2018). Not every participant is permitted to store the full data history by default and this provides added privacy by specifying which nodes are allowed to store sensitive data. Participants can only query data if they have permission to do so. Hyperledger Iroha has also introduced a new consensus algorithm based on the Byzantine fault tolerance algorithm known as Sumeragi (Bashir, 2017, p. 356). Iroha currently supports the platforms IOS, Android and JavaScript (Kedar Iyer, Rene Madsen, Solomon Lederer, Michael Wuehler, Joseph J. Bambara, 2018, p. 85).

3.5.2.4 Hyperledger Sawtooth Lake

Type: Permissioned, Public
Consensus: Proof-of-elapsed-time
Privacy: Intel SGX

Hyperledger Sawtooth Lake is a blockchain platform that supports both permissioned and permissionless modes, proposed by Intel in April 2016. (Bashir, 2017, p. 356). The Sawtooth Lake project is now fully open-sourced under the Linux Foundation umbrella project known as Hyperledger. Intel has also developed a novel consensus algorithm known as proof-of-elapsed-time (PoET), which makes use of Intel Software Guard Extension (SGX) architecture as a trusted execution environment (TEE) to ensure random and safe leader election (Baliga, 2017). One of the key features of Sawtooth Lake is that the transactions are decoupled from the ledgers ensuring flexibility for business processes utilizing transaction families (Bashir, 2017, p. 356). Transactions follow the patterns and structures defined in the transaction families. Sawtooth Lake's SDK supports multiple languages such as Go, JavaScript, Rust, C++ and Java (Saraf & Sabadra, 2018).

The Ethereum blockchain can be integrated with Sawtooth Lake via the integration project called, Seth. Seth enables developers to code smart contracts with Ethereum's smart contact language known as, Solidity (Saraf & Sabadra, 2018). This means that interoperability between Sawtooth and Ethereum is provided by Seth.

3.5.2.5 Hyperledger Indy

Type: Permissioned
Consensus: Redundant-BFT
Privacy: Zero-knowledge proofs

Indy was first introduced by the Sovrin foundation and is currently being developed by the Linux Foundation. The main focus of Hyperledger Indy is to provide a distributed ledger for digital identity management. This enables institutions to share a single user identity base. Users can provide consent to institutions to receive access to their common digital identity thus avoiding inconsistent and duplicate user records (Saraf & Sabadra, 2018). Redundant Byzantine fault tolerance (RBFT) is used by Hyperledger Indy as the consensus algorithm. The standard Byzantine fault tolerance algorithm is not considered by Indy to be effective enough. If the primary BFT node is a malicious node it could cause problems and possibly corrupt the network. RBFT is a novel approach inspired by Plenum, whose protocol ensures that malicious primary nodes can be detected (Saraf & Sabadra, 2018). The Linux Foundation has not yet released Indy as it is still in the incubation phase.

3.5.2.6 Hyperledger Tools

- **Cello**: Hyperledger Cello is a blockchain network provisioning toolkit (Kedar Iyer, Rene Madsen, Solomon Lederer, Michael Wuehler, Joseph J. Bambara, 2018, p. 195) that assists developers to easily deploy and maintain blockchain networks. In Hyperledger Fabric all the network artifacts and configuration files are set up manually. This process is time-consuming, challenging and prone to developer error. Each node in the network needs to be set up individually and terminated manually. Hyperledger Cello automates these tasks by automatically generating the network artifacts and configuration files (Kedar Iyer, Rene Madsen, Solomon Lederer, Michael Wuehler, Joseph J. Bambara, 2018, p. 195). These so-called artifacts are then automatically deployed to each node in the network. Developers only need to specify the topology of the blockchain network and Cello takes care of the rest.

- **Composer**: Hyperledger Composer is a set of tools for developing blockchain networks for business (Kedar Iyer, Rene Madsen, Solomon Lederer, Michael Wuehler, Joseph J. Bambara, 2018, p. 195). Composer simplifies the building of chaincode (Smart Contracts) with a model-based abstraction. Essentially, writing secure and robust chain-code in Fabric is more difficult and requires substantially more lines of code than writing the same chaincode in Composer (Kedar Iyer, Rene Madsen,

Solomon Lederer, Michael Wuehler, Joseph J. Bambara, 2018). This reduces the development time needed to produce robust chain-code by a significant factor.

Composer produces a Business Network Archive(.bna) file that consists of four types of files, namely Model(.cto), Scripts(js), Access Control(.acl) and Query(.qry) (Hyperledger, 2017). The Model(.cto) file describes the assets, participants, and transactions that will be used in the network and there can be multiple model files to keep related descriptions together. The Script(.js) file is where the business logic of the network is described in terms of transaction functions, which can be divided into different Script(.js) files to keep related functions together. The Access Control(.acl) file is where the rules of the network are described. For example, in which participants are permitted to control assets and execute transactions. The Query(.qry) file describes all the query definitions that might be used by participants on the network. These queries are written much like the views function in SQL. Therefore the Query(.qry) file only supports select statements. These files can be created by using the CLI Composer or the Composer Playground, which is a novel feature that enables developers to build and test their network definitions on the fly. This Business Network Archive file is deployed on top of the Fabric network. The Composer can easily be integrated with non-blockchain applications by exposing the network via a REST API (Hyperledger, 2017).

- **Explorer**: Hyperledger Explorer is a web-based tool that enables one to view the activities of deployed blockchain networks. Explorer displays blockchain details, for example, the number of peers, the number of blocks, the number of installed chain-codes and the number of transactions (Kedar Iyer, Rene Madsen, Solomon Lederer, Michael Wuehler, Joseph J. Bambara, 2018, p. 196). It is also possible to obtain more detail pertaining to each component, for example, one may view the details of any transaction.

3.5.3 Ethereum

Type: Public
Consensus: Proof-of-work, Proof-of-stake
Privacy: n/a

Ethereum was introduced in 2013 in a White Paper written by Vitalik Buterein, who was active in the Bitcoin community as a writer and programmer. He saw that Bitcoin had the potential to be much more than just a cryptocurrency. Buterein was interested in turning Bitcoin into a blockchain that could support business and government use. In this pursuit, he discovered that Bitcoin was not set up to handle the number of transactions a business use case would require. Bitcoin was already well established when Vitalik and numerous others realized that it would require a substantial code overhaul. The upgrade was considered too severe by the Bitcoin community. Vitalike and his team established the Ethereum Foundation in 2014 to raise funds for the development of a new blockchain protocol. Ethereum was first released in July 2015 Ethereum was first

released in July 2015 and is currently the most developed and innovative blockchain in use (Laurence, 2017, p. 42).

Ethereum has a cryptocurrency token known as Ether, which in Ethereum terms is the fuel of the network that is needed to execute code in the network (Bashir, 2017, p. 214). Ethereum has its own Turing-Complete programming language. This language enables developers to create programs, called smart contracts. Smart contracts are used to digitally facilitate, enforce or verify that all the terms, of a contract, are met before a transaction can take place (Bergquist, 2017). The smart contract eliminates the need for third parties involvement. Once a smart contract has executed it is tractable and irreversible. Thus the logic of a smart contract should be thoroughly tested before it is deployed on a production network (Bashir, 2017, p. 198). The smart contract feature of Ethereum is what makes it innovative and powerful. Ethereum provides developers with three ways to develop applications. Developers can make use of the main network which is public. All transactions are visible to all participants. The identity of participants is difficult to establish.

Developers can make use of the test network which is also public. The benefit of this network is that you can test your smart contracts, without using real cryptocurrency. The third method is to create a private network. This network is private to an extent. When a developer creates a network they are required to specify a network id and genesis block combination. This combination is used to identify and connect to the private network. The discomforting part of this type of network is that it is possible for the network id combination to be guessed (Bashir, 2017, p. 267). This makes it a real possibility for unauthorized participants to gain access to your so-called private network. Ethereum makes use of Proof-of-work consensus on their public networks (Anh et al., 2018). This type of consensus algorithms for business could be very expensive and slow. It does, however, provide a high degree of security in a public environment. Ethereum is currently working on a PoS consensus algorithm as an alternative to PoW (Bashir, 2017, p. 244).

3.5.4 Quorum

Type: Permissioned
Consensus: Raft, Istanbul-BFT
Privacy: Zero-knowledge proofs

The Quorum platform was developed by a finical institution known as J.P. Morgan Chase (Dagher, Mohler, Milojkovic, & Marella, 2018). Quorum is based on the Go implementation of the Ethereum protocol and therefore supports smart contracts written in Solidity. Quorum is classified as a permissioned blockchain network and the Quorum protocol supports two consensus algorithms known as Raft and Istanbul-BFT (J.P. Morgan Chase, 2018). Raft is used as the default consensus algorithm for Quorum (Anh et al., 2018). Quorum preserves data privacy through the use of segmentation, cryptogra-

phy and zero-knowledge proofs. Segmentation is used to divide each node's ledger into two segments, individually known as a public ledger and a private ledger. The quorum protocol introduced two novel features. namely private transactions and private smart contracts (J.P. Morgan Chase, 2018). Private transactions are used to send private data to a list of authorized nodes. These private transactions are stored in the authority node's private ledger segment. Zero-knowledge proofs are used to enhance the privacy of private transactions and cryptography is used to encrypt the private data flowing through the blockchain network. Cryptography is also used to generate a cryptographic hash of the encrypted data (J.P. Morgan Chase, 2018). Only the hash of the encrypted data is stored in the public ledger of the blockchain network (Dagher et al., 2018).

3.5.5 Parity Ethereum

Type: Private
Consensus: Proof-of-authority
Privacy: n/a

The Parity platform is based on the Go implementation of the Ethereum protocol. Parity makes use of a consensus algorithm known as proof-of-authority, which eliminates the need for mining and decreases transaction processing time by a significant factor. Proof-of-authority is considered more secure, as validating peer nodes need to be white-listed to join the network. This dramatically reduces the risk of attacks on the network. Only white-listed nodes can essentially modify blocks in the network. Parity is based on the Go implementation of Ethereum, also supports smart contracts written in Solidity, Serpent and LLL (Anh et al., 2018).

3.5.6 BigchainDB

Type: Public, Permissioned, Private
Consensus: Tendermint
Privacy: n/a

The BigchainDB platform was first introduced February 2016. BigchainDB is a distributed database with blockchain properties such as owner-controlled assets, decentralization and immutability (Gmbh, 2018). Traditional blockchain platforms are not good at storing and retrieving high volumes of data (Bashir, 2017, p. 42). The BigchainDB project strives to solve these issues by providing an immutable and tamper-resistant means of storage with a high query throughput. Traditional databases record data in tables but BigchainDB records data in the form of transactions (Gmbh, 2018). There are two kinds of transactions, namely, create transactions and transfer transactions. Create transactions are used to create immutable assets and transfer transactions are used to update and transfer ownership of assets. These transactions can be seen as a linked-list of objects each containing a value. The consensus algorithm currently being used by BigchainDB is Tendermint (Gmbh, 2018).

3.5.7 Corda

Type: Permissioned, Private
Consensus: Raft
Privacy: Zero-knowledge proof, State object

R3 first introduced the Corda platform on 30 November 2016 for the processing and recording of financial agreements (Brown, Carlyle, Grigg, & Hearn, 2016). The Corda platform is not considered as a blockchain platform in the traditional sense, (Bashir, 2017, p. 357) as traditional blockchains bundle transactions together into blocks that are then combined in a linked-list manner. Corda is a DLT platform that makes use of a hash tree instead of blocks to organize data (Bashir, 2017, p. 468). The Corda platform shares key features with traditional blockchain platforms, such as consensus, immutability, and authentication. Corda's architecture includes components such as state objects, contract code, legal prose, transactions, consensus, and flows. Smart contracts are used to create and manage the state objects and can be written in either Kotlin or JAVA (Anh et al., 2018). The state object is used as a digital document that contains a record of the content and current state of an agreement between parties (Brown et al., 2016). Transactions are used to alter the states of the state objects. Legal prose is encoded into the smart contracts' code to govern the agreement between the parties involved. Corda's consensus model is based on a notary service that is used to order and evaluate transaction uniqueness (Brown et al., 2016). There are multiple consensus algorithms that can be used by notaries like PBFT, Raft or Kafka (Brown et al., 2016). Corda has created a novel feature called flows that enables the development of decentralized work fowls (Bashir, 2017, p. 377). Flows run as an asynchronous state machine that interacts with users and nodes (Brown et al., 2016).

3.5.8 Ripple

Type: Permissioned
Consensus: Ripple consensus
Privacy: n/a

In 2012 Ripple Labs introduced a currency exchange and gross settlement platform named Ripple (Bashir, 2017, p. 288). Ripple is consensus-driven but is not considered as a blockchain platform in the traditional sense, the ripple protocol is just a simple permissioned DLT that makes use of a hash tree to organize transactions (Schwartz, Youngs, & Britto, 2014). Ripple Labs developed a consensus algorithm based on the Byzantine fault tolerance algorithm named the Ripple Protocol Consensus Algorithm (RPCA) (Schwartz et al., 2014). Each node in the Ripple network defines a unique node list (UNL), which is a list of trusted nodes that collect transactions in a data structure known as a candidate set (Baliga, 2017). These nodes broadcast their candidate sets to their UNL. Multiple rounds of votes are placed by nodes to validate each transaction in a candidate set and the candidate sets with the most votes progress to the next round of votes. This process continues until a candidate set achieves a supermajority vote of 80

percent of the nodes in the UNL. This candidate set with the supermajority vote is then considered as a valid block and added to the ledger forming a last closed ledger (LCL) in Ripple terms (Baliga, 2017). Ripple has received a virtual currency licensee from the New York Department of Financial Service (Kedar Iyer, Rene Madsen, Solomon Lederer, Michael Wuehler, Joseph J. Bambara, 2018, p. 22) and has its own cryptocurrency known as XRP. In 2017 XRP was considered the third-largest cryptocurrency in the world (Laurence, 2017, p. 135). The Ripple protocol is geared towards financial use such as inter-bank settlements (Bashir, 2017, p. 388).

3.5.9 Stellar

Type: Permissioned
Consensus: Stellar consensus
Privacy: n/a

Stellar is a permissioned distributed payment system that was released in 2015 to connect banks and people (Mazieres & Mazières, 2015). The Stellar project developed a novel consensus algorithm named the Stellar Consensus Protocol (SCP) (Kedar Iyer, Rene Madsen, Solomon Lederer, Michael Wuehler, Joseph J. Bambara, 2018, p. 215), which is based on the Federated Byzantine Agreement (FBA) algorithm (Mazieres & Mazières, 2015). This FBA forms groups of trusted parties known as quorums (Bashir, 2017, p. 393). The Stellar protocol has four main properties: decentralized control: enables any person to participate without having to work through a central authority; low latency: means higher transaction throughput, which is desired in business use cases; flexible trust: enables users to specify which parties they trust for any given reason and asymptotic security: achieved by using digital signatures and it has functions. The Stellar platform has its digital currency known as Lumens (XLM), which is required to execute transactions (Bashir, 2017, p. 394). Using Lumens (XLM) in this way mitigates denial of service attacks (Mazieres & Mazières, 2015).

3.5.10 Multichain

Type: Private
Consensus: Round-robin
Privacy: n/a

Multichain is a private blockchain platform adapted from the Bitcoin protocol (Kedar Iyer, Rene Madsen, Solomon Lederer, Michael Wuehler, Joseph J. Bambara, 2018). The Multichain platform was created for financial institutions to address the current problems facing the Bitcoin protocol, such as privacy, scalability and security. Multichain employs a round-robin like a consensus algorithm, the schedule of which is set by a parameter referred to as mining diversity. Each miner is enabled to create a number of blocks in a set period of time. Bitcoin's proof-of-work algorithm has been adapted to randomize the propagation of new blocks per miner. The dividends for transactions and mining

are zero by default. Multichain can support multiple cryptocurrencies and blockchain networks simultaneously, hence the name Multichain (Greenspan, 2013).

3.5.11 Kadena

Type: Private
Consensus: Scalable-BFT
Privacy: Onchain encryption

Kadena claims to have solved the scalability and privacy problems faced by blockchain platforms (Bashir, 2017, p. 384) by developing a proprietary consensus algorithm known as ScalableBFT (Anh et al., 2018). The ScalableBFT algorithm can scale thousands of nodes without experiencing significant performance fluctuations(Bashir, 2017, p. 384). Confidentiality of transactions is preserved with the use of on-chain symmetric encryption. Transactions can automatically be encrypted and decrypted by the participants in a transaction. Kadena also makes use of key rotation to prevent encryption keys from being compromised (Bashir, 2017, p. 384). Smart contracts can be coded in a language named Pact that was developed by Kadena. Kadena is a private blockchain that remains proprietary even though the Pact smart contract language has been open-sourced.

3.5.12 Tezos

Type: Public
Consensus: Proof-of-stake
Privacy: n/a

Tezos is a universal, automatic correction cryptocurrency ledger that supports the integration of cryptocurrencies such as Bitcoin, Ethereum and Cryptonotes (Goodman, 2016). The Tezos protocol strives to address the shortcomings of the Bitcoin protocol, such as problems incurred during hard forks, centralization of mining power and general security vulnerabilities (Bashir, 2017, p. 397). Ocaml is a functional programming language used in the development of Tezos (Goodman, 2016). The Tezos network employs a seed protocol that initiates the stakeholders of the network (Anh et al., 2018). Stakeholders are enabled to vote and approve amendments to the network. The seed protocol is based on the proof-of-stake consensus algorithm and is used instead of the traditional genesis block blockchain protocols (Goodman, 2016)

3.5.13 Dfinity

Type: Public
Consensus: Threshold relay
Privacy: n/a

Dfinity is a cloud computing platform that utilizes key features of blockchain technologies (*Dfinity Technology Overview Series, Consensus System*, 2018). The Dfinity protocol

is based on the Ethereum blockchain and the main goal is to create cloud 3.0 to run decentralized applications and services with on-chain governance. The architecture of Dfinity consists of four layers, namely the identity layer, the random beacon layer, the blockchain layer, and the notary layer (Anh et al., 2018). The responsibility of the identity layer is to register new clients that are required to make a security deposit to hold them accountable for their actions on the network (*Dfinity Technology Overview Series, Consensus System*, 2018). Dfinity makes use of a verifiable random function (VRF) to generate a random number referred to as a random beacon. The verifiable random function (VFR) is based on the threshold relay signature of the previous block that was created. The random beacon is used to select the next group of leader nodes. The leader nodes are enabled to create new blocks by signing them with a threshold signature (Anh et al., 2018). This process is known as the threshold relay and is based on the proof-of-stake consensus algorithm. The notary service layer is used to streamline the finality of transactions (*Dfinity Technology Overview Series, Consensus System*, 2018). Dfinity supports smart contracts written in Solidity, Serpent and LLL (Anh et al., 2018).

3.5.14 OpenChain

Type: Private
Consensus: Proof-of-authority
Privacy: n/a

Openchain is an open sourced private blockchain platform that is not strictly a blockchain, as it chains transactions directly together instead of grouping them in blocks. Chaining transaction directly together is inherently faster than grouping them in blocks first. Transactions in the Openchain network are linked immediately after being proposed to the network. Openchain thus enables real time approval of transactions and utilizes the proof-of-authority consensus algorithm (Anh et al., 2018). The Openchain network consists of a single trusted authority known as a validator and only the validator authority can append new blocks ("Privacy-preserving blockchain-based electric vehicle charging with dynamic tariff decisions", 2018). This leads to a single point of failure, which defeats the purpose of distributed networks. Smart contracts are not supported in Openchain. Openchain projects were abandoned when the company behind the projects closed down (Anh et al., 2018). The last commit on the GitHub repository https://github.com/openchain/openchain was made on 7 December 2016.

3.5.15 HydraChain

Type: Permissioned
Consensus: Tendermint based
Privacy: n/a

HydraChain is a permissioned protocol based on the Ethereum blockchain (Bashir, 2017, p. 397). Smart contracts are supported and can be coded in Solidity, Serpent and LLL (Anh et al., 2018). The consensus algorithm used by Hydrachain, as described in the documentation, is proprietary but based on the Tendermint protocol (Cachin & Vukolić, 2017). HydraChain seems to be an abandoned project as the last commit date in the GitHub repository (https://github.com/HydraChain/hydrachain) was made on 28 December 2016 (Anh et al., 2018). Thus there is a lack of viable documentation for the Hydrachain blockchain.

3.6 Comparison

This section highlights and compares the key features of each blockchain technology. These comparisons are presented in the form of tables, each of which focuses on various aspects of blockchain technologies. Table 3.1 compares various types of blockchain, namely public, permissioned and private. Public blockchain networks are open to anyone to participate at any level. Permissioned blockchain networks are mostly run by a consortium of organizations that need to share information in a trusted manner. The roles that an individual can perform in a permissioned network are controlled. Private blockchain networks are mostly owned and controlled by a single organization and membership in a private network is strictly controlled. Table 3.2 compares various consensus algorithms in terms of Byzantine fault tolerance, transaction speed, scalibility, and finality. Table 3.3 compares popular blockchain protocols with regards to the following criteria: blockchain type, consensus, data privacy, smart contract language, application and status.

Table 3.1: Comparison of blockchain types.

Criteria	**Public**	**Permissioned**	**Private**
Trust	Trustless	Semi-trusted	Trusted
Architecture	Decentralized	Moderately decentralized	Distributed
Immutability	Practically tamper-proof	Tamper-evident	Tamper-evident
Transparency	Full transparency	Semi-transparent	Semi-transparent, No trasnparency
Transaction Speed	Slow	Fast	Fast
Efficiency	Low	High	High

Table 3.2: Comparison of blockchain consensus algorithms.

Consensus	**Byzantine fault tolerance**	**Transactions per second**	**Scalibility**	**Finality**
Proof-of-work	✓	<100	High	Probabilistic
Proof-of-stake	✓	<1000	High	Probabilistic
Proof-of-authority	✓	#	High	Probabilistic
Proof-of-elapsed-time	✓	#	High	Probabilistic
Tendermint	✓	<=10k	Low	Deterministic
PBFT	✓	<2000	Low	Deterministic
Raft	x	>10k	Low	Deterministic
Kafka-ordering	x	-	Low	Deterministic
Ripple	✓	<1500	Low	Deterministic
Stellar	✓	<1000	Low	Deterministic
Sumeragi-BFT	✓	-	Low	Deterministic
Scalable-BFT	✓	-	Low	Deterministic

(✓) Supported; (x) Not-supported; (-) Unknown; (#) Depenends on implementation

Table 3.3: Comparison of popular blockchain protocols.

Platform	**Type**	**Consensus**	**Data Privacy**	**Smart Contract language**	**Application**	**Status**
Bitcoin	Public	Proof-of-work	-	Go, C++	Crypto-currency	Active
Ethereum	Public, Private	Proof-of-work, Proof-of-stake	-	Solidity, Serpent, LLL	Multi-purpose	Active
Parity Ethereum	Private	Proof-of-authority	-	Solidity, Serpent, LLL	Multi-purpose	Active
Quorum	Permissioned	Raft, Istanbul-BFT	ZKP	Solidity	Multi-purpose	Active
Hyperledegr Fabric	Permissioned, Private	Solo, Kafka	TLS, ZKP, Channels	Go, Java	Multi-purpose	Active
Hyperledegr Burrow	Permissioned	Tendermint	-	Solidity	Multi-purpose	Incubation
Hyperledger Sawtooth Lake	Permissioned, Public	Proof-of-elapsed -time	Intel SGX	Go, C++, etc.	Multi-purpose	Active
Hyperledger Iroha	Permissioned	Sumeragi-BFT	Channels	Java	Multi-purpose	Active
Hyperledger Indy	Permissioned	Redundant-BFT	ZKP	-	Decentralized identity	Incubation
R3 Corda	Permissioned, Private	Raft	ZKP, State object	Kotlin, Java	Financial	Active
Ripple	Permissioned	Ripple consensus	-	-	Financial	Active
Stellar	Permissioned	Stellar consensus	-	-	Financial	Active
BigchainDB	Public, Permissioned, Private	Tendermint	-	-	Multi-purpose	Active

Table 3.3: Comparison of popular blockchain protocols. (continued)

Platform	**Type**	**Consensus**	**Data Privacy**	**Smart Contract language**	**Application**	**Status**
Multichain	Private	Round-robin	-	C++	Financial, Crypto-currency	Active
Dfinity	Public	Threshold relay	-	Solidity, Serpent, LLL	Multi-purpose	Incubation
Kadena	Private	Scalable-BFT	Onchain encryption	Pact	Multi-purpose	Active
Tezos	Public	Proof-of-stake	-	Michaleson	Michaleson applications	Active
Openchain	Private	Proof-of-authority	-	-	Financial	Inactive
Hydrachain	Permissioned	Tendermint based	-	Solidity, Serpent, LLL	Multi-purpose	Inactive

3.7 Conclusion

This chapter served to address secondary objective two by outlining the components and features of various blockchain technologies suitable for storing and sharing of information. The chapter firstly identified that there are three types of blockchain networks namely, public, permissioned and private (Bashir, 2017, p. 25). These network types are suited for different use-cases. Consensus algorithms are one of the key components of blockchain networks. There are mainly two types of consensus algorithms proof-based and Byzantine fault tolerance based (Bashir, 2017, p. 28). Various consensus algorithms were discussed such as Proof-of-work, Proof-of-stake, Proof-of-authority, Proof-elapsed-time, Practical byzantine fault tolerance, Raft, Kafka ordering, and Tendermint. Each of these consensus algorithms provides unique characteristics suited for different use-cases. Blockchain technology has been evolving over the years and introduces features such as Smart Contracts, private and confidential transactions. These features make blockchain technology more suitable for enterprise use-cases beyond financial and crypto-currency (Bashir, 2017, p. 473). Smart contracts are used to digitally facilitate, enforce and verify that all the terms, of a contract (business logic), are met before a transaction can take place (Bergquist, 2017). One of the major challenges for the realization of enterprise-grade blockchain technology is privacy (Bashir, 2017, p. 473). Various privacy algorithms have been discussed promising enhanced privacy for blockchain technology. Popular blockchain technology protocols have been explored and contrasted in the form of tables.

Chapter 4

Initial Report

4.1 Introduction

Chapter 2 served as a study of relevant literature to ascertain the current state of electronic health records and the South African policies surrounding them. The current state of blockchain technology was discussed in Chapter 3 and Chapter 4 serves as an initial report covering the identified challenges and solutions about electronic health records systems. Key challenges were identified from the review of relevant literature

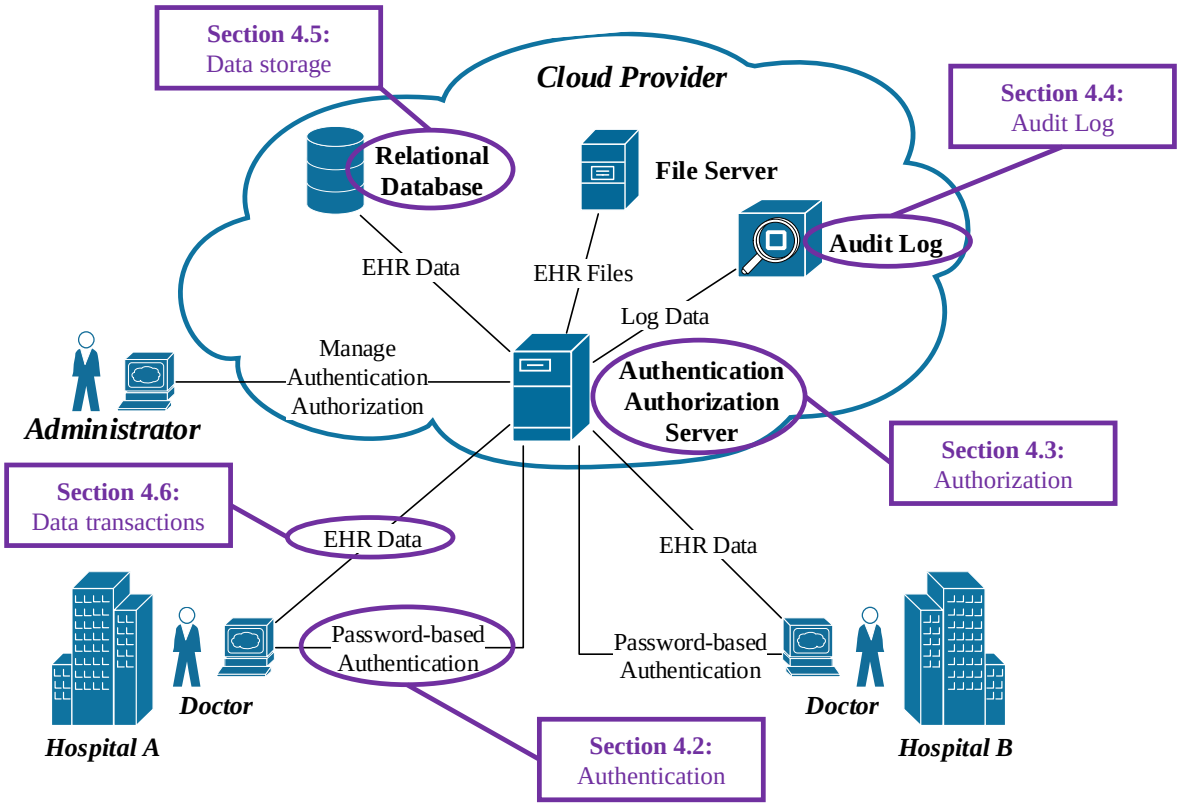

Figure 4.1: EHR system overview with blockchain intervention diagram

and these challenges could be addressed with aspects of blockchain technologies. Figure 4.1 provides an overview diagram of the current model used by electronic health record systems as well as where blockchain technology might improve the model.

4.2 Authentication

Authentication is used to identify a user requesting access to a system. It is important for systems to be able to identify users in order to restrict access to a system functions. Users identities are also required for audit purposes (Cilliers, 2017). The electronic health record systems authentication process has been identified as a weak point. Blockchain technology could improve this model by utilizing techniques found in Chapter 3.

4.2.1 Current state of EHR systems

Electronic health record (EHR) systems make use of password-based authentication (Kshetri, 2018). The password-based authentication model makes use of a user ID and password combination to authenticate users. Password-based authentication used in a centralized architecture is considered vulnerable to cyber-attacks (Mosakheil, 2018). Intuitions often make use of insecure clouds to store passwords. These passwords are also, more often than not, considered to be weak and easily crack-able by utilizing techniques such as social engineering, password guessing, and brute-forcing. Furthermore, passwords are also being reused for different service which makes it easier to exploit systems utilizing the same passwords (Kshetri, 2017). Password-based authentication, on its own, is no longer considered secure.

4.2.2 Possible blockchain intervention

Public key infrastructure (PKI) certificate-based authentication could improve the current EHR systems' authentication process (Kshetri, 2018). Permissioned blockchain networks currently make use of PKI to authenticate and identify users in the blockchain network. This type of authentication is known as certificate-based authentication (Emmadi et al., 2019). The users' public key and identity are encoded into a digital certificate and the PKI certificate-based authentication process is considered to be more secure than password-based authentication (Kshetri, 2018). PKI certificate-based authentication could improve the current electronic health record systems' authentication process.

4.3 Authorisation

Authorization is used to determine the actions an authenticated user can perform on a system (Cilliers, 2017). System functions should be restricted with the use of appropriate authorization mechanisms and only users that need access to particular functions should be able to access them. Restricting rights in this manner mitigate the risk of unauthorized disclosure of information (Seol et al., 2018). The electronic health record systems'

authorization process has been identified as a weak point that blockchain technology could improve.

4.3.1 Current state of EHR systems

Electronic health record (EHR) systems commonly make use of a role-based access control model to enforce authorization (Seol et al., 2018). This control model provides authorization to users based on the roles assigned to them. A role is generally determined by the job function of a user, such as a doctor would have the role of doctor assigned to them. Each role is assigned access rights based on a need to know basis (Pussewalage & Oleshchuk, 2016). These roles are generally statically assigned by the system administrator (Bokefode et al., 2014). The role-based model does not support dynamic attributes, for example, control of user permissions based on the time of day. The current EHR systems require a fine-grained, distributed and dynamic access control mechanism (Seol et al., 2018).

4.3.2 Possible blockchain intervention

Attribute-based access control (ABAC) could be used to improve the EHR systems' authorization process by providing a more fine-grained and dynamic authorization mechanism. Permissioned blockchain technology utilizes the ABAC model in which the access control rules are embedded in the code of the smart contract. If the access control rules need to be changed, the code of the smart contract must be modified and redistributed to the blockchain network (Emmadi et al., 2019). Permissioned blockchain technology could thus provide an improved authorization model for EHR systems based on ABAC.

4.4 Audit log

An audit log is a recording of all the actions a user performs on a system. Audit logs are useful for identifying how, when, where, why and by whom data was accessed, modified or leaked. Criminals frequently tamper with audit logs to cover their tracks (Dekker & Etalle, 2007) and the audit log model used by EHR systems is particularly vulnerable due to a single point of failure, as illustrated in Figure 4.1. Blockchain technology could provide an improved audit log model.

4.4.1 Current state of EHR systems

Electronic health record systems do not generate standardized audit logs but their audit logs are stored in a centralized architecture, which may present a single point of failure. Electronic healthcare records are often subjected to tampering for several reasons that include insurance fraud and various criminal offenses (Kshetri, 2018). Data integrity cannot be assured without an immutable audit log and it is therefore important that

EHR systems keep an immutable audit log to identify criminals tampering with healthcare records. Audit logs have a key role in identifying culprits responsible for data breaches (Kshetri, 2017).

4.4.2 Possible blockchain intervention

Permissioned blockchain technology can be used to generate a semi-decentralized, tamper-evident and standardized audit log for electronic health record systems. Permissioned blockchain networks make use of membership service to identify users interacting in the blockchain network (Bashir, 2017, p. 362). The user's identity can be used to record all the actions performed by that user in the blockchain network. Blockchain technology provides a tamper-evident and peer-to-peer means of storage and this storage model can ensure data integrity from data creation to data retrieval. Permissioned blockchain technology can utilize techniques such as zero-knowledge proofs and channels to enhance the privacy of users' personal information throughout the audit process. Zero-knowledge proofs can be used to hide the user's private information whilst still proving that the information is accurate and in this manner permissioned blockchain technology could ensure a balance between transparency and privacy in the electronic health records audit process (Bünz et al., 2017).

4.5 Data storage

Data storage is the act of recording information electronically and this can be achieved by utilizing various structures and architectures. All of the available structures and architectures have both advantages and disadvantages. Centralized data storage, for example, a relational database used by EHR systems, provides a high degree of transaction throughput but could be vulnerable due to a single point failure (Liang et al., 2017). Blockchain technology could be used to improve the current centralized model used by EHR systems.

4.5.1 Current state of EHR systems

Healthcare institutions traditionally make use of a client-server model to maintain their electronic records, which are generally stored in a centralized architecture such as a relational database, which represents a single point of failure (Liang et al., 2017). Electronic healthcare records are often subject to tampering for various reasons that include insurance fraud and criminal offenses. Healthcare providers also often make use of potentially insecure clouds to store shared secrets. The U.S. health insurer, Anthem's, data breach that occurred in December 2014 exposed more than 80 million clients' sensitive information (Kshetri, 2018). This attack is an example of why centralized architectures are often considered easy and lucrative targets.

4.5.2 Possible blockchain intervention

Blockchain technology could provide a peer-to-peer, decentralized and tamper-evident means of append-only storage for electronic healthcare record systems (Bashir, 2017, p. 438). This storage model can ensure data integrity from data creation to data retrieval with a single point of failure being averted by using blockchain technology (Kshetri, 2018). Permissioned blockchain technology can be used to enhance the privacy of data being stored in a blockchain network. Cryptographic techniques such as zero-knowledge proofs can be used to store data privately and ensure data integrity without revealing private information (Bünz et al., 2017). Permissioned blockchain technology could thus improve the current storage architecture in use by EHR systems.

4.6 Data transactions

Transactions are used to add, update or retrieve data from databases as well as share information between authorized parties. The transaction process is used by the EHR system's work technology but data integrity, privacy, and policy-related concerns need to be addressed (Catalini, 2017). Blockchain technology might be able to address some of these concerns.

4.6.1 Current state of EHR systems

Healthcare institutions make use of three models when sharing electronic healthcare records, namely the push, the pull and the view models (Catalini, 2017). The traditional models in use lack a standardized means of generating an audit trail (Kshetri, 2017) and therefore cannot ensure data integrity from data creation to data use and retrieval (Catalini, 2017). Audit trails perform a key role in identifying culprits responsible for data breaches. There are also data privacy issues surrounding these models, although they are technologically sound. The consent process used by these models is traditionally handled in an informal ad-hoc fashion but time constraints could undermine the quality of this process. Patients might consent to the disclosure of information without fully understanding how their personal information will be disclosed and processed. Patients have the right to stipulate with whom their information is shared but numerous healthcare institutions lack the resources or capabilities to store patients' consent stipulations (Kshetri, 2017).

4.6.2 Possible blockchain intervention

Blockchain technology could improve the EHR systems data transactions process by providing a secure peer-to-peer means of transferring information (Bashir, 2017, p. 438). Data integrity could be ensured from data origin to data retrieval with the use of a tamper-evident audit log (Kshetri, 2017). Permissioned blockchain technology could enhance the privacy of data transactions through the use of cryptography techniques

such as Zero-knowledge proofs and channels (Hyperledger, 2019). With the use of permissioned blockchain technology, patients could be able to stipulate with whom their information can be shared as well as what aspects of information will be shared (Kshetri, 2017).

4.7 Conclusion

This chapter served as an initial report. Various challenges and solutions to the current state of EHR systems were discussed. The initial report serves as a theoretical overview of the possible solutions blockchain technology could provide for shortcomings in EHR systems. These theories are evaluated through experimentation in Chapter 5.

Chapter 5

Blockchain feasibility testing

5.1 Introduction

This chapter outlines the experimental process and possible blockchain interventions related to electronic health record systems. Blockchain technologies are suited to a variety of cases and only a select few were utilized for experimentation, namely Hyperledger Fabric, Ethereum, Quorum, and Hyperledger Indy. The various experiments that were conducted revealed that Hyperledger Fabric could be a suitable candidate for illustrating the benefits that blockchain technology could provide to EHR systems and only those experiments conducted with Hyperledger Fabric were included. Hyperledger Fabric is a flexible general-purpose blockchain that has been built with privacy and confidentiality in mind (Anh et al., 2018). The Fabric project provides build-in features that could be used for different use-cases. Furthermore, the modular approach followed by the Fabric project enables developers to create custom features to address their specific requirements. Thus, Hyperledger Fabric appears to be the best platform for any use-case that requires privacy and flexibility. Even though this chapter only covers an implementation with Hyperledger Fabric it does not mean that Fabric is the only permissioned blockchain technology capable of improving EHRs. The possible solutions illustrated in this chapter adhere to the policies and regulations identified in Chapter 2. This chapter serves to address secondary objective three which aims to conduct experiments with aspects of blockchain technologies, for the secure storage and distribution of electronic health records.

5.2 Experimental process

This section outlines the experimental process that was followed. Firstly, the documentation for each blockchain technology was read to establish suitable use cases. The blockchain technologies that could be suited to address EHR challenges were selected and these were explored through their respective getting started examples. Example applications (smart contracts) were deployed and tested on the example blockchain network. These examples were then dissected to understand the inner workings of the

blockchain network setup. The experimental approach was systematic and logical.

5.2.1 Documentation

Reading the documentation of each blockchain technology was the first step in the experimental process. Documentation performs a vital role in understanding and developing blockchain solutions and key areas were explored when reading the documentation. These explored areas included: suitable use cases; key features; privacy features; network setup and smart contacts SDK/API. These areas were revisited throughout the experimental process.

5.2.2 Experimental notes

Notes were made throughout the experimental process. These notes recorded details such as the network setup; the process followed; the problems encountered and the solutions to the problems. These notes were used to reflect on the possible solutions that blockchain technology could provide to EHR systems.

5.2.3 Network setup

The blockchain network setup forms the foundation of the blockchain solution. Blockchain technologies normally provided a *quick start* network setup, which enables developers to learn by example. The first step in the experimental network setup is to deploy and test the *quick start* network setup provided and secondly to explore the running *quick start* network and configuration files to understand how the blockchain network functions. The next step is to create a blockchain network from the ground up with the support of the documentation and the *quick start* network setup provided. When the experimental network is fully functional, key features are enabled and explored. These network features include privacy and security enhancements such as network isolation and Transport Layer Security (TLS), which mitigates the risk of eavesdropping, tampering and the forging of data packets traveling over the network. Wireshark is also used to validate and explore these network features. Wireshark is an open-source network packet analyzer (Orebaugh, Ramirez, & Beale, 2006, p. 53). It should also be noted that the experimental network setup is first tested on a single machine and then distributed to multiple machines. This is done to minimize the problems that could be encountered during network deployment. The experimental network is then ready to run applications known as smart contracts.

5.2.4 Application setup

Blockchain technologies usually include an example of smart contracts. These examples assist developers to deploy and test smart contracts with minimal investment. The first step in the experimental application process is to deploy a sample smart contract to the experimental network that has been created. The smart contract's methods and

functions are then tested. If these functions work as expected the sample smart contract is dissected to understand the components and structures involved. Experimental smart contracts are developed from the ground up with the support of the sample smart contracts and the API/SDK documentation. These features are validated by exploring the blocks created in the blockchain network. Wireshark is also used to evaluate the packets sent on the network when transactions are executed.

5.2.5 Docker virtualization

Docker is an open-source platform for operating system-level virtualization in the form of containers (McKendrick & Gallagher, 2017, p. 9). It was first released in 2013 by the Docker Corporation. Docker performs an instrumental role in this experimental process. The Docker platform is a prerequisite for developing with blockchain technology, such as Hyperledger Fabric, Hyperledger Indy and more. Docker can also be used with any software technology to streamline and package solutions into containers. The Docker platform enables developers to create, build and use images, containers, networks, and volumes. A Docker image can be seen as a read-only instruction set for the creation of containers. The images contain instructions about the operating system to use, the software to install and the setup procedures for these components. Docker images can also extend other images created. Containers are the running version of an image. Docker containers store and share data with the host machine and other containers through the use of volumes. The docker network function enables containers to communicate with other containers and machines. Each network component used in this experimental process was provided in the form of a Docker image. Those that were not provided as a docker image were configured and built into a Docker image.

5.2.5.1 Docker compose

Docker-compose is a tool that enables developers to manage the deployment of multiple containers in the form of application services, which are configured with a YAML file. The YAML files used by Docker compose are generally named in the format docker-compose.yaml. The docker-compose.yaml file can comprise multiple containers and their configuration parameters, such as environment variables, volumes, networks and which ports to expose to the host machine. The experimental network components used in this study were all grouped into different docker-compose.yaml files. Refer to Code snippet 5.1 for an example of a docker-compose.yaml file. This enables developers to test applications on their development machines and deploy them to any machine running the Docker engine. Using Docker in this manner enables the experiments to be reproduced and tested in a variety of scenarios.

```
version: '3.2'

networks:
  hospital-network:
    external: true

services:
  ca.hospitala.org.com:
    container_name: ca.hospitala.org.com
    image: hyperledger/fabric-ca:1.4.0
    environment:
      - CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=hospital-network
    ports:
      - "7054:7054"
    command: sh -c 'fabric-ca-server start -b user:pass -d'
    volumes:
      - ./crypto-config/peerOrganizations/hospitala.org.com
    networks:
      - hospital-network
    restart: always
```

Code snippet 5.1: docker-compose.yaml

5.3 Hyperledger Fabric experimental network

Hyperledger Fabric consists of multiple network components known as clients, peers, certificate authorities (CA), orderers, zookeeper clusters and Kafka clusters, as illustrated in Figure 5.1. The client nodes enable users to interact with the Hyperledger Fabric network. Hyperldeger Fabric provides a configurable Fabric-Client. Hyperldeger Fabric also enables developers to create a custom client through the use of the Fabric-SDK provided. The certificate authority is responsible for issuing, validating and revoking digital certificates. The peers are the nodes in the network that store the linked list of blocks. The orders are responsible for ensuring that the blocks are valid and propagated in the correct sequence. Fabric enables two modes of ordering, namely Solo and Kafka. Solo ordering comprises a single orderer and is not meant for production use. The Kafka ordering mode is used for production. The ordering service, including the zookeeper and Kafka cluster, make up the consensus protocol for Hyperledger Fabric. The zookeeper cluster is responsible for maintaining the state of the Kafka cluster and the Kafka cluster is responsible for keeping the organization's orderers in sync. Each organization's network, for example, Hospital B, will comprise their individual clients, peers, orderer, and certificate authority network components. The diagram in Figure 5.1 illustrates a network with only two hospitals. The actual experimental network consists of three hospitals. The diagram purposely excludes the third hospital as it will needlessly complicate the diagram without adding value.

5.3.1 Network features

Hyperledger Fabric's goal is to provide a platform for organizations to share information in a privacy-preserving environment. Fabric enables developers to represent participants as organizations. An organization in this experimental network is defined as a hospital and doctors work under the authority of hospitals. The constructed network consists of three organizations known as Hospital A, Hospital B, and Hospital C. Fabric channels enable isolation between organizations. This experimental network makes use of three channels to simulate and test the isolation between the hospitals. Organizations that are part of a channel can send private transactions to one another. These private transactions will not appear on any organizational ledger outside the channel. A channel could be viewed as a separate blockchain that begins with two blocks. The first block is referred to as the genesis block and the second as the channel configuration block. The channel configuration block contains a membership service provider (MSP), policies and a list of organizational peers authorized to receive channel updates. Channel policies are used to establish the rules of engagement. An example policy could state that a transaction needs to be signed by each organization's administrator for the transaction to be valid. Mutual TLS has also been enabled in all of the network components to mitigate the risk of a man in the middle attack. Hyperledger Fabric's smart contracts could also be used to create and maintain private data collections, which are collections of data shared between the organization's part of a channel. Private data collections are stored and shared outside the blockchain. Only the proof of the data collection is stored in the blockchain for audit purposes. Fabric's private data collection enables private data to only be accessible by authorized parties for a given period of time. The data is purged after the specified time has elapsed. The proof of the transaction remains in the blockchain for audit purposes.

5.3.2 Network configuration files

Hyperledger Fabric mainly utilises two configuration files, namely *crypto-config.yaml* and *configtx.yaml*. The *crypto-config.yaml* file is used as a bootstrap network configuration. Fabric developed a tool known as Cryptogen that is able to interpret the *crypto-config.yaml* file. Code snippet 5.2 contains code from the *crypto-config.yaml* file used in this experimental setup. The output of the Cryptogen tool is a set of digital certificates and private keys for each network component. The Cryptogen tool was intended to be used for testing environments, as the digital certificates and private keys are generated in a static manner. The production process could include using the Fabric-CA to dynamically generate digital certificates and private keys. The *configtx.yaml* file is used to specify the properties of the consensus protocol, genesis file and channel artifacts to be used by the blockchain network. Code snippet 5.3 contains code from the *configtx.yaml* file used in this experimental setup. The Fabric Configtxgen tool is used to generate these artifacts for the blockchain network.

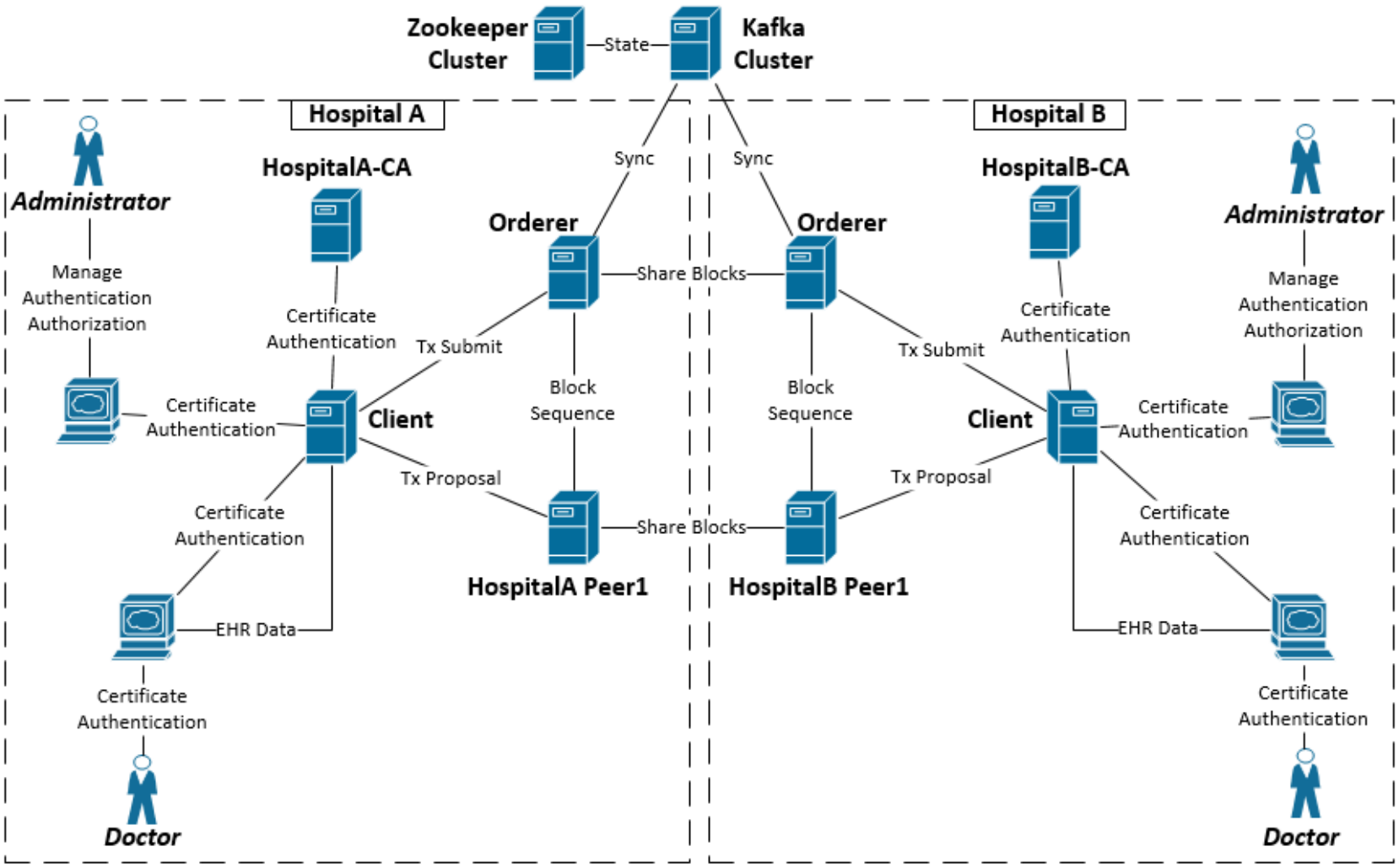

Figure 5.1: Experimental Hyperledger Fabric network diagram

```
OrdererOrgs:
  - Name: Orderer
    Domain: org.com
    Specs:
      - Hostname: orderer0.hospitala
      - Hostname: orderer0.hospitalb
      - Hostname: orderer0.hospitalc

PeerOrgs:
  - Name: Hospitala
    Domain: hospitala.org.com
    EnableNodeOUs: true
    Template:
      Count: 4 //number of peers
    Users:
      Count: 4 //number of users

  - Name: HospitalB
    Domain: hospitalb.org.com
    EnableNodeOUs: true
    Template:
      Count: 4
    Users:
      Count: 4

  - Name: HospitalC
    Domain: hospitalc.org.com
    EnableNodeOUs: true
    Template:
      Count: 4
    Users:
      Count: 4
```

Code snippet 5.2: crypto-config.yaml

```
Organizations:
    - &OrdererOrg
        Name: OrdererOrg
        ID: OrdererMSP
        MSPDir: crypto-config/ordererOrganizations/org.com/msp

    - &HospitalA
        Name: HospitalaMSP
        ID: HospitalaMSP
        MSPDir: crypto-config/peerOrganizations/hospitala.org.com/msp
        AnchorPeers:
            - Host: peer0.hospitala.org.com
              Port: 7051

    - &HospitalB
        Name: HospitalbMSP
```

Code snippet 5.3: configtx.yaml

```
        ID: HospitalbMSP
        MSPDir: crypto-config/peerOrganizations/hospitalb.org.com/msp
        AnchorPeers:
            - Host: peer0.hospitalb.org.com
              Port: 7051

    - &HospitalC
        Name: HospitalcMSP
        ID: HospitalcMSP
        MSPDir: crypto-config/peerOrganizations/hospitalC.org.com/msp
        AnchorPeers:
            - Host: peer0.hospitalc.org.com
              Port: 7051

Capabilities:
    Global: &ChannelCapabilities
        incompatibilities.  Users
        V1_1: true

    Orderer: &OrdererCapabilities
        V1_1: true

    Application: &ApplicationCapabilities
        V1_2: true

Application: &ApplicationDefaults
    Organizations:

Orderer: &OrdererDefaults
    OrdererType: kafka

    Addresses:
        - orderer0.hospitala.org.com:7050
        - orderer0.hospitalb.org.com:8050
        - orderer0.hospitalc.org.com:9050
    BatchTimeout: 2s
    BatchSize:
        MaxMessageCount: 10
        AbsoluteMaxBytes: 99 MB
        PreferredMaxBytes: 512 KB

    Kafka:
        Brokers:
            - kafka0.org.com:9092
            - kafka1.org.com:10092
            - kafka2.org.com:11092
```

Code snippet 5.3: configtx.yaml (continued)

```
    Organizations:

Profiles:
    OrderersHospitalGenesis:
        Capabilities:
            <<: *ChannelCapabilities
        Orderer:
            <<: *OrdererDefaults
            Organizations:
                - *OrdererOrg
            Capabilities:
                <<: *OrdererCapabilities
        Consortiums:
            HospitalConsortium:
                Organizations:
                    - *HospitalA
                    - *HospitalB
                    - *HospitalC

    Comunitychannel:
        Consortium: HospitalConsortium
        Application:
            <<: *ApplicationDefaults
            Organizations:
                - *HospitalA
                - *HospitalB
                - *HospitalC
            Capabilities:
                <<: *ApplicationCapabilities

    HospitalABchannel:
        Consortium: HospitalConsortium
        Application:
            <<: *ApplicationDefaults
            Organizations:
                - *HospitalA
                - *HospitalB
            Capabilities:
                <<: *ApplicationCapabilities

    HospitalACchannel:
        Consortium: HospitalConsortium
        Application:
            <<: *ApplicationDefaults
            Organizations:
                - *HospitalA
                - *HospitalC
            Capabilities:
                <<: *ApplicationCapabilities
```

Code snippet 5.3: configtx.yaml (continued)

5.3.3 Deploying the network

The first step in deploying the blockchain network is to create the cryptographic material and the channel artifacts. This can be achieved by using the Cryptogen and Configtxgen tools, as discussed in Section 5.3.2. Once all of the network materials have been created, a docker-compose.yaml file should be created for each organization, for example, docker-compose-hospitala.yaml. This file should contain all of Hospital A's network components. The blockchain network is then deployed by executing the Docker command to bring up the blockchain network, for example, docker-compose -f docker-compose-hospitala.yaml up -d. The network component logs should be inspected to make sure the network has been deployed successfully. When the network is up and running the Fabric-client is used to manually create and join organizational peers to their respective channels. The peers should be inspected to ensure that they have successfully joined the channels. In the preceding sections, individual experiments were outlined in terms of providing possible solutions to the challenges discussed in Chapter 4. The experiments that follow are Experiment 1: authentication; Experiment 2: authorization; Experiment 3: audit log; Experiment 4: storage; Experiment 5: data transactions.

5.4 Experiment 1: Authentication

Authentication is used to identify a user requesting access to a system. It is important for systems to be able to identify users in order to restrict access to a system's functions. Users' identities are also required for audit purposes (Cilliers, 2017). The electronic health record system's authentication process has been identified as a weak point and Blockchain technology could improve this model by utilizing techniques found in Chapter 3.

5.4.1 Current state of EHR systems

Electronic health record (EHR) systems make use of password-based authentication (Kshetri, 2018). The password-based authentication model makes use of a userID and password combination to authenticate users. Password-based authentication used in a centralized architecture is considered vulnerable to cyber-attacks (Mosakheil, 2018) and institutions often make use of insecure clouds to store passwords. These passwords are also, more often than not, considered to be weak and easily cracked by utilizing techniques such as social engineering, password guessing, and brute-forcing. Passwords are also often reused for various services, which makes it easier to exploit systems utilizing the same passwords (Kshetri, 2017). Password-based authentication, on its own, is no longer considered secure.

5.4.2 Possible blockchain intervention

Public key infrastructure (PKI) certificate-based authentication could improve the current EHR system's authentication process (Kshetri, 2018). Permissioned blockchain net-

works currently make use of PKI to authenticate and identify users in the blockchain network. This type of authentication is known as certificate-based authentication (Emmadi et al., 2019). The user's public key and identity are encoded into a digital certificate, which makes the PKI certificate-based authentication process more secure than password-based authentication (Kshetri, 2018). PKI certificate-based authentication could thus improve the current electronic health record system's authentication process.

5.4.3 Hyperledger Fabric

Hyperledger Fabric could be used to improve the EHR's current authentication process. The experiments conducted with Hyperledger Fabric revealed that it can be used to issue and revoke digital certificates. Only valid digital certificates can access the blockchain network. When a certificate is revoked, the owner of that certificate can no longer use it to access the network. The valid certificate contains the identity of a user interacting with the blockchain network. It is thus possible to log the actions and events triggered by a user. This section illustrates the steps followed to reproduce this experiment.

5.4.3.1 Authentication experiment

This experiment aimed to test certificate-based authentication with Hyperledger Fabric. The Fabric experimental blockchain network should first be brought online and tested. The administrator account should be able to register users with a one time password on the Fabric CA. The user should be able to employ their one-time password to enroll with the Fabric CA, which should issue a digital certificate and private key pair to each user. The digital certificate serves as the user's identity. Chaincode has been developed containing methods such as GetID(), GetMSP() and GetCertificate(). The developed chaincode should be deployed to the blockchain network. Refer to Code snippet 5.4 for an example of a method used to gain a user's identity. One of the user accounts should be used to invoke the chaincode methods using a transaction proposal. The result of the invocation should be the user's identity. The administrator's account should be able to revoke the user's digital certificate from the blockchain network by updating the certificate revocation list (CRL) with the Fabric CA. The revoked user account should not be able to invoke chaincode. The request to invoke chaincode should be denied, as the digital certificate has been revoked.

```
func (s *SmartContract) getUserIdentitty(APIstub shim.
    ChaincodeStubInterface, args []string) sc.Response {

    //Get user identity
    userID, err := cid.GetID(APIstub)
    if err != nil {
      return shim.Error("Could not get user identity")
    }

    //Get user certificate
    cert, err := cid.GetX509Certificate(APIstub)
    if err != nil {
      return shim.Error("Could not get user certificate")
    }

    //Get membership service id
    msp, err := cid.GetMSPID(APIstub)
    if err != nil {
      return shim.Error("Could not get user msp")
    }

    owner := Owner{ID: userID, Cert: cert, Msp: msp}
    ownerAsBytes, _ := json.Marshal(owner)

    return shim.Success(ownerAsBytes)
}
```

Code snippet 5.4: getUserIdentitty method

5.4.3.2 Results of the experiment

The administrator account was the only user able to register users with the Fabric CA. User accounts could be enrolled only once utilizing their corresponding one time password. The user's digital certificates were able to execute chaincode in the blockchain network. The response of the transaction was the user's identity, digital certificate, and membership service identity. Digital certificates can be revoked by the administrator account. Revoked digital certificates were not able to access the blockchain network.

5.4.3.3 Discussion

History dictates that password-based authentication is not sufficiently secure for sensitive information. The human factor in password-based authentication remains a weak point. Passwords are often created by humans and are thus usually short and weak. This is because it is difficult for humans to remember long and complex passwords. In 2019 a strong password is considered to be one that contains a mixture of lowercase letters, uppercase letters, numbers, special characters and is more than eight characters in length. An example of a password would typically be the user's pet's name followed by a special character followed by their birth year, such as Maya@2008. According to the howsecureismypassword.net website, the Maya@2008 password could be cracked in

as little as four weeks without knowing anything about the user (Collider, 2019). Social engineering could be applied and the password could be cracked in a matter of days. Digital certificates are created by computers that understand the importance of entropy. Certificates in 2019 are created with a key size of 2048bits, which translates to 256 characters. This would take a standard computer quadrillion years to crack. Digital certificates come with two keys known as a private key and a public key. The public key is derived from the private key and the public key is used to encrypt a message that can only be decrypted with the corresponding private key. Messages can also be signed with the use of the private key and can be verified with the corresponding public key. Both keys are thus required to impersonate an individual. The advantages and disadvantages of the solution are discussed further in terms of the storage process, the authentication process, and the duration of exposure.

Storage process

Passwords are commonly stored in a centralized database. When an authentication database is compromised, passwords can be stolen. These stolen passwords can lead to data breaches that could lead to many more data breaches. The private keys of certificates are commonly stored in the user's machine. Users are encouraged to safeguard their private keys by storing them in a hardware security module (HSM) or a trusted platform module (TPM). These security measures are also flawed but that is not discussed in this section. Hackers need to attack a user directly to steal their private keys. The data exposure from such an attack leads to minor data exposure depending on the user's access level. These attacks can also be traced more directly.

Authentication process

The authentication server's TLS mechanisms have been compromised, resulting in an insecure channel of communication. When a user authenticates using a password that password travels over the network as plain text. This opens the door for a man in the middle attack. The user's password has now been stolen. When a user authenticates with certificate-based authentication only the public key is exposed over the wire. An attacker would need both the public and private keys to impersonate the user. Cracking the public key would not be practical, as previously discussed.

Exposure duration

Passwords are often used across various systems, rarely expire and new passwords are often based on the old passwords. Certificates come with an expiration date, which could, for example, be set according to the user's privilege levels. This implies that if a certificate is compromised it can only be used for a limited period. Stolen certificates can also be revoked by adding them to the certificate revocation list (CRL).

5.5 Experiment 2: Authorization

Authorization is used to determine the actions an authenticated user can perform on a system (Cilliers, 2017). System functions should be restricted with the use of appropriate authorization mechanisms and only users that need access to functions should be able to gain access. Restricting rights in this manner mitigate the risk of unauthorized disclosure of information (Seol et al., 2018). The electronic health record system's authorization process has been identified as a weak point that blockchain technology could improve.

5.5.1 Current state of EHR systems

Electronic health record (EHR) systems commonly make use of a role-based access control model to enforce authorization (Seol et al., 2018). The role-based access control model provides authorization to users based on the roles assigned to them. A role is generally determined by the user's job function, for example, doctors would have the role doctor assigned to them. Each role is assigned access rights on a need to know basis (Pussewalage & Oleshchuk, 2016). These roles are generally statically assigned by the system administrator (Bokefode et al., 2014). The role-based model does not support dynamic attributes, for example, control of user permissions based on the time of day. The EHR systems of today need a fine-grained, distributed and dynamic access control mechanism (Seol et al., 2018).

5.5.2 Possible blockchain intervention

Attribute-based access control (ABAC) could be used to improve EHR systems authorization process by providing a more fine-grained and dynamic authorization mechanism. Permissioned blockchain technology makes use of the ABAC model. The access control rules are embedded in the code of the smart contract. If the access control rules need to be changed, the code of the smart contract needs to be modified and redistributed to the blockchain network (Emmadi et al., 2019). Permissioned blockchain technology could thus provide an improved authorization model for EHR systems based on ABAC.

5.5.3 Hyperledger Fabric

Hyperledger Fabric could be used to improve the EHR system's authorization process. The experiments conducted with Hyperledger Fabric revealed that it can be used to restrict access to the system based on a user's attributes. Fabric encodes these attributes into the digital certificate and only users that are assigned specific attributes can access methods that require the same set of attributes. These attributes can also be revoked at any time, which would effectively terminate a user's access to specific functions. It is thus possible to restrict access to functions based on a set of attributes. This section illustrates the steps followed to reproduce this experiment.

5.5.3.1 Authorization experiment

The authorization experiment aimed to test attribute-based access control ABAC with Hyperledger Fabric. The administrator should be able to register a set of users with various attributes and the users should then be able to enroll with the Fabric-CA and receive a transaction certificate (TCert), which contains the user's authorization attributes. Chaincode has been installed with attribute-based access control rules at a method level. This TCert can now be used to authenticate and authorize a user on the system. Only authenticated and authorized users should be able to execute chaincode methods. When a user is not authorized to execute methods an error message should be displayed. Blockchain technology allows the restriction of access to specific records based on a user's identity. Policy requirements could state that doctors need consent from patients to access their records. These kinds of policies can be encoded into the EHR's chaincode and only doctors with consent would be able to gain access to patients' EHRs. Refer to Code snippet 5.5 for an example of a method to gain access to a patient's health record. This experiment allows patients to add doctors to their list of authorized doctors. Refer to Code snippet 5.6 for an example of a method used to add a doctor to a patient's authorization list. Doctors that are not on this list should not be able to gain access to the patient's record. Policies can, however, be implemented to override the authorization mechanism in the event of a patient being declared incapacitated by a doctor. When a user is no longer authorized to access methods, a CRL could be generated and propagated over the network. This would effectively terminate the user's access to the system.

```
func (s *SmartContract) getPatientEHR(APIstub shim.ChaincodeStubInterface,
    args []string) sc.Response {

    //Get user identity
    if len(args) != 1 {
        return shim.Error(argsErr)
    }

    //Get user identity
    userID, err := cid.GetID(APIstub)
    if err != nil {
        return shim.Error(idenitiyErr)
    }

    //Check if user has attribute
    err := cid.AssertAttributeValue(APIstub,"doctor","true")
    if err != null {
        return shim.Error(authErr)
    }
```

Code snippet 5.5: getPatientEHR method

```
    patientAsBytes, _ := APIstub.GetState(args[0])
    patient := Patient{}
    json.Unmarshal(patientAsBytes, &patient)

    //Check if doctor is on the list
    val, hasKey := patient.AuthDoctor[userID]
    if hasKey != true {
        return shim.Error(authErr)
    }

    return shim.Success(patientAsBytes)
}
```

Code snippet 5.5: getPatientEHR method (continued)

```
func (s *SmartContract) addDoctorToPatient(APIstub shim.
    ChaincodeStubInterface, args []string) sc.Response {

    //Get user identity
    if len(args) != 1 {
        return shim.Error(argsErr)
    }

    //Get user identity
    userID, err := cid.GetID(APIstub)
    if err != nil {
        return shim.Error(idenitiyErr)
    }

    //Check if user has attribute
    err :=cid.AssertAttributeValue(APIstub,"patient","true")
    if err != null {
        return shim.Error(authErr)
    }

    doctor := Doctor{}
    doctorAsBytes, _ := APIstub.GetState(args[0])
    json.Unmarshal(doctorAsBytes, &doctor)

    patient := Patient{}
    patientAsBytes, _ := APIstub.GetState(userID)
    json.Unmarshal(patientAsBytes, &patient)

    patient.AuthDoctor[doctor.DoctorID] = doctor
    patientAsBytes, _ = json.Marshal(patient)
    api.PutState(userID, patientAsBytes)

    return shim.Success(patientAsBytes)
}
```

Code snippet 5.6: addDoctorToPatient method

5.5.3.2 Results of the experiment

The administrator was able to register a user with a set of attributes. Users can successfully enroll with the Fabric-CA and receive their respective TCerts with attributes. The users that did not have the appropriate attributes could not access the chaincode methods. The users with the correct set of attributes could gain access to these methods. The user Doctor1 was present on the patient's list and could, therefore, gain access to the patient's EHR. Doctor2 was denied access to a patient's EHR although Doctor2 did have access to the chaincode methods. This is because Doctor2 did not have consent from the patient to view their EHR. Doctor2 requested consent from the patient and the patient added Doctor2 to their approved list. Doctor2 could then gain access to the patient's EHR. The administrator revoked attributes by effectively terminating the user's certificate by generating a CRL. This CRL was propagated over the network resulting in the user being denied access to the system. The user had to re-enroll with the Fabric-CA to receive a new certificate with a new set of attributes.

5.5.3.3 Discussion

Role-based access control (RBAC) is a popular and effective access control model with limitations. RBAC assigns roles to users based on their job function and every role can be seen as a group of restrictions and privileges. These groups are generally based on organizational policies. The role of doctors, for example, groups all doctors in the policy group of a doctor. The role doctor, therefore, grants all doctors the same privileges. When there is a sub-group of doctors, a new role group needs to be created. Role-based access control is simple and easily maintained within small organizations but large organizations often need complex roles over time, resulting in role explosion. It becomes difficult to implement and maintain thousands of roles. When there is a need for a more fine-grained and complex access control model then ABAC is a better approach. The advantages and disadvantages of the solution are discussed in terms of the storage process, the access control rules, the access control audit and access control granularity.

Storage process

The role-based access control rules are commonly encoded into the application code that is hosted on a centralized server. When a user requests access to resources, the user requires a specific role, which is generally stored and mapped in a centralized relational database that may present a single point of failure. If the relational database is compromised, an attacker could add or remove privileges or hijack the highest privileged user account. This could result in unauthorized access to the system. The ABAC attributes are encrypted into digital certificates that are stored on the user's machine, the security aspects of which were discussed in Section 5.4. The access control rules are encoded into the chaincode that is distributed to each peer in the network and an attacker would need to either steal an administrator's certificate or compromise the majority of peers in the network. These two actions are considered to be unpractical

and attacks such as these could easily be detected and traced, as each action in the blockchain network is recorded.

Access control rules

Role-based access control (RBAC) can only define rules with static parameters and rule parameters, therefore, need to be known by the system before it is deployed. Attribute-based access control (ABAC), on the other hand, can make use of dynamically defined attributes that are evaluated at run-time. An example is that a rule could state that specific doctors may only access the system during working hours. The system would check the time of day at run-time before access is granted. Attributes could support various complex organizational rules and policies. ABAC is thus more flexible and can support complex rules and policies.

Access control audit

The attribute-based access control (ABAC) model could be difficult and time-consuming to implement. The ABAC model might also make access control auditing difficult, as each individual is often assigned a different set of attributes. Access to specific data objects can be restricted to dynamic parameters but an auditor would need to map out the individual attributes related to each object or data access rule. RBAC is more simple, as the auditor would only need to follow the permissions of a role group and not an individual's attributes related to a specific data rule.

Access control granularity

Role-based access control (RBAC) can only restrict access to specific actions based on the user's role. Each customized user role would require a new role per custom attribute. Over time this could result in role explosion. Role explosion si when the number of roles exceeds the number of users in a system. The RBAC model is thus not suitable for restricting access to individual operations or specific data objects. Attribute-based access control (ABAC) enables organizations to implement a granular and fine-grained access control model and access can effectively be restricted based on a set of four attribute types. These attribute types are commonly known as subject attributes, action attributes, object attributes, and contextual attributes. Subject attributes are related to the user requesting access to the system, e.g., department, role, age, etc. Action attributes describe the actions a user is allowed to perform on a system, e.g., write, read, delete, etc. Object attributes are used to define the object types a user is authorized to access, e.g., department, bank account, medical record, etc. Contextual attributes deal with the dynamic aspects of access control, such as the user's location, the time of day, etc. Combinations of these attribute types can support significantly complex and fine-grained access control rules.

5.6 Experiment 3: Audit log

An audit log is a recording of all the actions a user performed on a system. Audit logs are use-full in identifying how, when, where, why and by who data was accessed, modified and leaked. Tampering with audit logs frequently occur to cover a criminal's tracks (Dekker & Etalle, 2007). The audit log model used by EHR systems is vulnerable due to a single point of failure as illustrated in Figure 4.1. Blockchain technology could provide an improved Audit log model.

5.6.1 Current state of EHR systems

Electronic health record systems do not generate standardized audit logs. These audit logs are also stored in a centralized architecture that may present a single point of failure. Electronic healthcare records are often subjected to tampering for various reasons, namely insurance fraud, criminal offenses, and more (Kshetri, 2018). Data integrity cannot be assured without an immutable audit log and it is therefore important that EHR systems maintain an immutable audit log to identify criminals tampering with healthcare records. Audit logs have a key role in identifying culprits responsible for data breaches (Kshetri, 2017).

5.6.2 Possible blockchain intervention

Permissioned blockchain technology can be used to generate a semi-decentralized, tamper-evident and standardized audit log for electronic health record systems. Permissioned blockchain networks make use of membership service to identify users interacting in the blockchain network (Bashir, 2017, p. 362). The user's identity can be used to record all the actions performed by that user on the blockchain network. Blockchain technology provides a tamper-evident and peer-to-peer means of storage, which can ensure data integrity from data creation to data retrieval. Permissioned blockchain technology can utilize techniques such as zero-knowledge proofs and channels to enhance the privacy of users' personal information throughout the audit process. Zero-knowledge proofs can be used to hide the user's private information whilst still proving that the information is accurate (Bünz et al., 2017). Permissioned blockchain technology could thus ensure a balance between transparency and privacy in the electronic health records audit process.

5.6.3 Hyperledger Fabric

Hyperledger Fabric could be used to improve the EHR system's audit log process. The experiments conducted with Hyperledger Fabric revealed that it can be used to record a tamper-evident audit log. Fabric utilizes digital certificates to identify users interacting with the system. There are two aspects to the audit logs about Fabric. The linked-list of blocks containing a set of transactions serves as an automatic audit log and changes made to a specific record can be retrieved over time with the corresponding record key. Chaincode could be developed to record actions performed on the network and using

chaincode for audit log purposes provides developers with flexibility, as each audit log entry is encoded into a transaction. Changes made to the audit log record are also automatically recorded over time. The concept of Fabric channels could enhance the privacy of organizational audit logs. Chaincode can be deployed on different channels and only organizations that are part of a channel can effectively interact with chaincode on that channel. This could be a useful feature, as audit logs can be isolated with the use of channels. Zero-knowledge proofs can be used to hide users' identities and data from an auditor.

5.6.3.1 Audit log experiment

The aim of the audit log experiment was to test an audit log created with Hyperledger Fabric. The audit log in this experiment was used to track the changes to a record over time and to keep a history of actions performed by authorised users on the network. The administrator should be able to register user accounts for auditors and the auditors should be able to enroll with Fabric-CA and receive their digital certificate. Audit log chaincode was deployed to the experimental network. Audit log entries can be appended with the *addAuditLogEntry()* method, refer to Code snippet 5.7. The audit log chaincode is used to record the user's interactions with a patient's health record. It should be noted that the hash of the modified patient's record is also stored for integrity purposes. Auditors are only permitted to access methods that require the auditor attributes. Auditors can execute methods such as *getAuditLogEntry()*, *getHistoryForAuditKey()* and *getAuditLogByRange()*. The *getAuditLogEntry()* method illustrated in Code snippet 5.8 returns a single audit log entry. The *getHistoryForAuditKey()* method illustrated in Code snippet 5.9 returns the changes made to a specific record over time. The *getAuditLogByRange()* method illustrated in Code snippet 5.10 returns the audit log by range.

```
func (s *SmartContract) addAuditLogEntry(APIstub shim.
    ChaincodeStubInterface, args []string) sc.Response {

    //Expecting elven arguments
    if len(args) != 11 {
      return shim.Error("Incorrect number of arguments.")
    }

    txntmsp, err := APIstub.GetTxTimestamp()
    if err != nil {
      return shim.Error("Error converting timestamp")
    }
    time := time.Unix(txntmsp.Seconds, int64(txntmsp.Nanos)).String()

    auditLog := AuditLog{}
    auditLog.LogID = args[0]
    auditLog.Timestamp = time
```

Code snippet 5.7: addAuditLogEntry method

```
    auditLog.UserIdentity = args[1]
    auditLog.UserRole = args[2]
    auditLog.MSP = args[3]
    auditLog.Channel = args[4]
    auditLog.ChaincodeVersion = args[5]
    auditLog.ChaincodeName = args[6]
    auditLog.FunctionCall = args[7]
    auditLog.RecordID = args[8]
    auditLog.RecordHash = args[9]
    auditLog.Status = args[10]

    auditLogAsBytes, _ := json.Marshal(auditLog)
    APIstub.PutState(args[0], auditLogAsBytes)

    return shim.Success(nil)
}
```

Code snippet 5.7: addAuditLogEntry method (continued)

```
func (s *SmartContract) getAuditLogEntry(APIstub shim.
    ChaincodeStubInterface, args []string) sc.Response {

    //Expecting single argument
    if len(args) != 1 {
        return shim.Error("Incorrect number of arguments.")
    }

    //Check if user has attribute
    err :=cid.AssertAttributeValue(api,"auditor","true")
    if err != null {
    return shim.Error(authErr)
    }

    auditLogAsBytes, _ := APIstub.GetState(args[0])
    return shim.Success(auditLogAsBytes)
}
```

Code snippet 5.8: getAuditLogEntry method

```
func (s *SmartContract) getHistoryForAuditKey(APIstub shim.
    ChaincodeStubInterface, args []string) sc.Response {
    //Expecting single argument
    if len(args) != 1 {
    return shim.Error("Incorrect number of arguments.")
    }

    //Check if user has attribute
    err :=cid.AssertAttributeValue(api,"auditor","true")
    if err != null {
        return shim.Error(authErr)
    }
```

Code snippet 5.9: getHistoryForAuditKey method

```
    resultsIterator, err := APIstub.GetHistoryForKey(args[0])
    if err != nil {
      return shim.Error(err.Error())
    }

    defer resultsIterator.Close()
    buffer := buildTimeLine(resultsIterator)
    buffer.String())

    return shim.Success(buffer.Bytes())
}
```

Code snippet 5.9: getHistoryForAuditKey method (continued)

```
func (s *SmartContract) getAuditLogByRange(APIstub shim.
    ChaincodeStubInterface, args []string) sc.Response {

    //Expecting two arguments
    if len(args) != 2 {
      return shim.Error("Incorrect number of arguments.")
    }

    //Check if user has attribute
    err :=cid.AssertAttributeValue(api,"auditor","true")
    if err != null {
        return shim.Error(authErr)
    }

    resultsIterator, err := APIstub.GetStateByRange(args[0], args[1])
    if err != nil {
      return shim.Error(err.Error())
    }

    defer resultsIterator.Close()
    buffer := buildStateTimeLine(resultsIterator)
    buffer.String())

    return shim.Success(buffer.Bytes())
}
```

Code snippet 5.10: getAuditLogByRange method

5.6.3.2 Results of the experiment

The administrator was able to register accounts for auditors with a set of attributes and the auditors could successfully enroll with the Fabric-CA. The actions performed by doctors and patients were successfully recorded in the blockchain network and the auditors were only able to gain access to the functions they were authorized to execute. The *getAuditLogEnry()* method successfully returned an audit log record. When an audit log entry was modified it was automatically recorded as a transaction in the network.

The transactions associated with the audit log key were traversed and returned with the *getHistoryForAuditKey()* method. The *getAuditLogByRange* method successfully returned a range of audit log entries.

5.6.3.3 Discussion

Audit logs are responsible for recording every action performed on a system. The audit log is used for audit purposes and could be used to track data breaches. Traditional audit log systems are built around a centralized architecture. Audit logs are often stored in a relational database or on a file server. While these methods of storage work practically it does, however, represent a single point of failure. Compromising one of these storage methods would enable cyber-criminal to erase their tracks. Therefore, actions performed by cyber-criminals could go undetected. As a result, the integrity of the data stored in the relational database cannot be guaranteed. Blockchain-based audit logs are practically permanent and tamper-evident. Since blockchain is an append-only data structure that is distributed across several peers. Cyber-criminals would have to attack the majority of the peers in the network simultaneously to corrupted the audit log. This attack would not go unnoticed. Even if cyber-criminals can hijack a user account the changes made by the account would not go undetected. The changes made to the audit log would be appended leaving the previous records intact. This could then be used to flag suspicious accounts and track the cyber-criminals responsible. Therefore, data integrity can be preserved through the use of blockchain technology.

5.7 Experiment 4: Data storage

Data storage is the act of recording information electronically. Data can be stored utilizing different structures and architectures. All of the structures and architectures come with advantages and disadvantages. Centralized data storage such as a relational database used by EHR systems provide a high degree of transaction throughput but could be vulnerable due to a single point failure (Liang et al., 2017). Blockchain technology could be used to improve the current centralized model used by EHR systems.

5.7.1 Current state of EHR systems

Healthcare institutions traditionally employ a client-server model to maintain their electronic health records. These records are generally stored in a centralized architecture such as a relational database, which represents a single point of failure (Liang et al., 2017). Electronic healthcare records are often subject to tampering for various reasons, which include insurance coverage, criminal offenses and more (Kshetri, 2018). Healthcare providers also often make use of potentially insecure clouds to store shared secrets. The U.S. health insurer Anthem's data breach, which occurred in December 2014 exposed more than 80 million clients' sensitive information (Kshetri, 2018). This attack is an example of why centralized architectures are often considered easy and lucrative targets.

5.7.2 Possible blockchain intervention

Blockchain technology could provide a peer-to-peer, decentralized and tamper-evident means of append-only storage for electronic health record systems (Bashir, 2017, p. 438). This storage model can ensure data integrity from data creation to data retrieval and a single point of failure can be averted with the use of blockchain technology (Kshetri, 2018). Permissioned blockchain technology can be used to enhance the privacy of data being stored in a blockchain network. Cryptographic techniques such as zero-knowledge proofs can be used to store data privately and ensure data integrity without revealing private information (Bünz et al., 2017). Permissioned blockchain technology could thus improve the current storage architecture in use by EHR systems.

5.7.3 Hyperledger Fabric

Hyperledger Fabric could be used to improve the EHR storage process. The experiments conducted with Hyperledger Fabric revealed that it could be used to record electronic health records in a privacy-preserving manner. Channels could be used to isolate EHR data between organizations and only organizations that are part of a channel would be able to interact with the stored data. The practically immutable append-only data structure provided by Hyperledger Fabric preserves data integrity and Fabric could thus be used to improve the EHR storage mechanism.

5.7.3.1 Storage experiment

The aim of the storage experiment is to test the storage capabilities of blockchain technology. The experiment also aims to test the privacy features provided by Hyperledger Fabric. The first step of the experiment was to create and deploy a PatientEHR smart contract. The PatientEHR smart contract contains the following methods, *addPatientEHR()*, *updatePatientEHR()*, *GetPatientEHR()*, *GetPatientEHRHistory()*. The smart contract should be deployed across multiple channels to test the isolation of data between channels. Different EHRs should be added to each channel via the *addPatientEHR()* method refer to Code snippet 5.11. When the *getPatientEHR()* method as illustrated is Code snippet 5.5 executed on a channel it should only retrieve the records stored on that channel. An electronic health record can be updated with the use of the *updatePatientEHR()* method refer to Code snippet 5.12. Once an EHR record has been updated the data is appended to that specific record. The *getPatientEHRHistory()* method as illustrated by Code snippet 5.13 should retrieve all the modifications made to an EHR over time. This method would reveal how, when and by whom the patient's electronic health record has been modified.

```
func (s *SmartContract) addPatientEHR(APIstub shim.ChaincodeStubInterface,
    args []string) sc.Response {

    if len(args) != 5 {
      return shim.Error("Incorrect number of arguments")
    }

    //Get user identity
    userID, err := cid.GetID(APIstub)
    if err != nil {
      return shim.Error("Could not get user identity")
    }

    //Check if user has attribute
    err :=cid.AssertAttributeValue(api,"doctor","true")
    if err != null {
        return shim.Error(authErr)
    }
    patient := Patient{
        PatientID:         args[0],
        PatientName:       args[1],
        PatientSurname:    args[2],
        PatientFileURL:    args[3],
        PatientRecordHash: args[4],
    }
    doctorAsBytes, _ := APIstub.GetState(userID)
    doctor := Doctor{}
    json.Unmarshal(doctorAsBytes, &doctor)

    patient.AuthDoctor = make(map[string]Doctor)
    patient.AuthDoctor[userID] = doctor
    patientAsBytes, _ := json.Marshal(patient)
    APIstub.PutState(args[0], patientAsBytes)

    auditLog := AuditLog{}
    auditLog.LogID = patient.PatientID
    auditLog.UserIdentity = userID
    auditLog.UserRole = "doctor"
    auditLog.ChaincodeName = "patient-ehr"
    auditLog.Channel = "communitychannel"
    auditLog.ChaincodeVersion = "v1.0"
    auditLog.FunctionCall = "addPatient"
    auditLog.RecordHash = patient.PatientRecordHash
    s.AuditLog(APIstub, auditLog)

    return shim.Success(nil)
}
```

Code snippet 5.11: addPatientEHR method

```
func (s *SmartContract) updatePatient(APIstub shim.ChaincodeStubInterface,
    args []string) sc.Response {

    //Expecting four arguments
    if len(args) != 4 {
      return shim.Error("Incorrect number of arguments.")
    }

    //Get user identity
    userID, err := cid.GetID(APIstub)
    if err != nil {
      return shim.Error("Could not get user identity")
    }

    //Check if user has attribute
    err :=cid.AssertAttributeValue(api,"doctor","true")
    if err != null {
        return shim.Error(authErr)
    }

    patientAsBytes, _ := APIstub.GetState(args[0])
    patient := Patient{}
    json.Unmarshal(patientAsBytes, &patient)

    //Check if doctor is on the list
    val, hasKey := patient.AuthDoctor[userID]
    if hasKey != true {
        return shim.Error("Not authorized to update patient's record")
    }

    patient.PatientName = args[1]
    patient.PatientSurname = args[2]
    patient.PatientFileURL = args[3]
    patientAsBytes, _ = json.Marshal(patient)
    APIstub.PutState(args[0], patientAsBytes)

    auditLog := AuditLog{}
    auditLog.LogID = patient.PatientID
    auditLog.UserIdentity = userID
    auditLog.UserRole = "doctor"
    auditLog.ChaincodeName = "patient-ehr"
    auditLog.Channel = "communitychannel"
    auditLog.ChaincodeVersion = "v1.0"
    auditLog.FunctionCall = "updatePatient"
    auditLog.RecordHash = patient.PatientRecordHash
    s.AuditLog(APIstub, auditLog)

    return shim.Success(nil)
}
```

Code snippet 5.12: updatePatientEHR method

```
func (s *SmartContract) getHistoryForPatient(APIstub shim.
    ChaincodeStubInterface, args []string) sc.Response {

    //Expecting single argument
    if len(args) != 1 {
      return shim.Error("Incorrect number of arguments.")
    }

    //Get user identity
    userID, err := cid.GetID(APIstub)
    if err != nil {
      return shim.Error("Could not get user identity")
    }

    //Check if user has attribute
    err :=cid.AssertAttributeValue(api,"doctor","true")
    if err != null {
        return shim.Error(authErr)
    }

    patientAsBytes, _ := APIstub.GetState(args[0])
    patient := Patient{}
    json.Unmarshal(patientAsBytes, &patient)

    //Check if doctor is on the list
    val, hasKey := patient.AuthDoctor[userID]
    if hasKey != true {
        return shim.Error("Not authorized to update patient's record")
    }

    resultsIterator, err := APIstub.GetHistoryForKey(args[0])

    if err != nil {
        return shim.Error(err.Error())
    }

    defer resultsIterator.Close()
    buffer := buildTimeLine(resultsIterator)

    return shim.Success(buffer.Bytes())
}
```

Code snippet 5.13: getHistoryForPatient method

5.7.3.2 Results of the experiment

The chaincode was successfully deployed over multiple channels. Running the *addPatientEHR()* method on each channel was executed successfully. When retrieving an EHR record from a channel it was clear that each channel stored data in an isolated fashion. Modifying a patient's EHR by running the *updatePatientEHR()* method was executed successfully. When retrieving the information with the *getPatientEHR()* only

the updated version of the patient's record was retrieved. When running the *getPatientEHRHistory()* method it was clear that the modifications made to an EHR were stored and tampering with EHR records could thus be detected.

5.7.3.3 Discussion

Electronic health records are currently stored in a centralized architecture such as a relational database that may present a single point of failure. The peer-to-peer nature of blockchain technology could avert a single point of failure. Relational databases are commonly hosted by a cloud provider, which is cost-effective, scalable and flexible. This method of storage works well from an accessibility standpoint, as the electronic health records are all stored in the cloud. However, the increased accessibility has created challenges like security and privacy issues. The results of data breaches to EHRs serves as proof. The advantages and disadvantages of the solution are discussed further in terms of data ownership, availability, scalability and EHR policies.

Data ownership

The fact that EHRs make use of cloud providers raises the question of ownership. Who owns the data stored on the cloud service? The healthcare providers would like to believe that they are the data owners but cloud providers own the physical hard drives on which the information is stored. This would suggest that the data physically belongs to the cloud provider. The healthcare providers have no control over how the data is physically distributed and policies only protect healthcare providers. Data breaches mostly affect healthcare providers financially and they could recover their financial losses. Patients, on the other hand, cannot recover their reputation. Data leaked as a result of a data breach cannot be unleaked. Financial compensation could never fully restore a patient's reputation. Blockchain technology could enable healthcare providers to become the owners of the records and ensure the same level of accessibility. Blockchain technology could also enable patients to become owners of their electronic health records. The practicality of patient-centric EHRs is currently being researched.

Availability

Cloud providers commonly guarantee a service level agreement (SLA) of 99% uptime. What happens when the cloud services temporally go offline? Healthcare providers would temporally lose access to data. Furthermore, an internet connection is required to gain access to cloud services and internet service providers could also go offline resulting in availability issues that could have unintended consequences. Blockchain technology could improve the availability of electronic health records, as data is stored peer-to-peer with blockchain technology. If the internet goes offline, healthcare providers would still have access to their copy of the blockchain data. Healthcare providers would thus have access to the version of the EHRs that existed before the loss of internet connectivity.

Scalability

Healthcare providers generally store more than 20 terabytes of data annually in both text and picture format. Cloud providers periodically increase data storage. Economies of scale enable cloud providers to scale large organizations at low premiums and the cloud infrastructure could thus easily scale to accommodate the rate at which EHRs grow. Blockchain technology does not deal well with large file sizes, such as pictures and some researchers suggest that large files should be stored outside the blockchain. Only a reference to the large files should be stored on the blockchain network.

EHR Policies

Storing electronic health records on a blockchain network is mostly in line with the policies surrounding EHRs. The HPCSA stipulates that when health records are stored in an electronic format it should be done so in an append-only format. Copies of the records should be made and stored in different physical locations. The copies are then used to detect tampering with the records. Health records should also be kept for at least five years. Blockchain technology is aligned with these policies, as the data stored in a blockchain network is distributed across peer nodes situated in different physical locations. The nature of blockchain technology is to store records permanently in an append-only format. Relational databases are not in line with these policies, as the data is not stored in an append-only format. The POPI act, however, states that users should be able to request that their personally identifiable information (PII) is purged from a service (South Africa, 2013, section 24). Since blockchain technology stores data permanently, it cannot comply with this requirement. Relational databases satisfy this requirement by supporting the purging of records. Even if the policies are adapted for the storage of EHRs, there remains a problem with storing information encrypted in a blockchain network. Cryptographic algorithms come with a shelf life. The data stored on the blockchain cannot be re-encrypted with a new algorithm and encrypted information could thus be compromised if an attacker gains access to the blockchain network after the fact. Researchers propose storing cryptographic hashes of data on the blockchain for integrity and audit purposes. Thus, blockchain technology could be used to store pointers to EHRs and hashes of EHRs to ensure the integrity of the records.

5.8 Experiment 5: Data transactions

Transactions are used to add, update or retrieve data from databases. In this context data transactions are also the sharing of information between authorized parties. The transaction process is used by the EHR system's work technology but there are data integrity, privacy, and policy-related concerns that blockchain technology might be able to address (Catalini, 2017).

5.8.1 Current state of EHR systems

Healthcare intuitions utilize three models when they share electronic health records, namely the push, the pull, and the view models (Catalini, 2017). The traditional models in use lack a standardized means of generating an audit trail (Kshetri, 2017) and therefore cannot ensure data integrity from data creation to data use (Catalini, 2017). Audit trails have a significant role in identifying culprits responsible for data breaches. There are also data privacy issues surrounding these models, although they do work technologically. The consent process used by these models is traditionally handled in an informal, ad-hoc fashion. Time constraints could undermine the quality of this process and patients might consent to the disclosure of information without fully understanding how their personal information will be disclosed and processed. Patients have the right to stipulate with whom their information is shared but numerous healthcare intuitions lack the resources or capabilities to store patients' consent stipulations (Kshetri, 2017).

5.8.2 Possible blockchain intervention

Blockchain technology could improve the EHR system's data transactions process by providing a secure peer-to-peer means of transferring information (Bashir, 2017, p. 438). Data integrity could be ensured from data origin to data retrieval with the use of a tamper-evident audit log (Kshetri, 2017). Permissioned blockchain technology could enhance the privacy of data transactions through the use of cryptography techniques such as zero-knowledge proofs and channels (Hyperledger, 2019). With the use of permissioned blockchain technology, patients could be able to stipulate with whom their information can be shared as well as what aspects of information may be shared (Kshetri, 2017).

5.8.3 Hyperledger Fabric

The experiments conducted with Hyperledger Fabric revealed that it could be used to transact electronic health records in a privacy-preserving manner. Private data collection could be used to share private data between a sub-group of an organization's part of a channel. The private data collection is updated with the use of private transactions, which are confidential between participants and anonymous to normal users. The private transaction occurs on the physical blockchain and only proof of the private transactions is stored on the blockchain for audit purposes. These private transactions can be tracked through the use of zero-knowledge proofs, thus effectively preserving patients' privacy and ensuring accountability through the use of the zero-knowledge proofs.

5.8.3.1 Data transaction experiment

The data transaction experiment aimed to test the transacting capabilities of blockchain technology. The first step in the experimental process was to develop a smart contract with a private data collection. The private data collection should be set up to transact

data between Hospital A and Hospital B. The private data should also be set to automatically purge after a specified period. Refer to Code snippet 5.14 for an example of a *private-data-collection.json* configuration file. The second step was to deploy the smart contract on the community channel. Refer to Code snippet 5.15 for an example of a method to transact a patient's private data. The private data collection should only be shared between two organizations. The organization not a part of the private data collection should not be able to retrieve the private data. Refer to Code snippet 5.16 for an example of a method to obtain a patient's private data. The information should no longer be available after the period specified in the chaincode.

```
[
    {
        //Collection name
        "name": "private_data",
        //Share data bertween Hospital A and B.
        "policy": "OR('HospitalaMSP.member', 'HospitalbMSP.member')",
        "requiredPeerCount": 1,
        "maxPeerCount": 3,
        "blockToLive":5, //Set time period for data purge
        "memberOnlyRead": true
    }

]
```

Code snippet 5.14: private-data-collection.json

```
func (s *SmartContract) trasactPatientPrivateData(APIstub shim.
   ChaincodeStubInterface) sc.Response {

    //Get user identity
    userID, err := cid.GetID(APIstub)
    if err != nil {
        return shim.Error("Could not get user identity")
    }

    //Check if user has attribute
    err :=cid.AssertAttributeValue(APIstub,"doctor","true")
    if err != null {
        return shim.Error(authErr)
    }

    transientMap, err := APIstub.GetTransient()

    if _, ok := transientMap["private_data"]; !ok {
      return shim.Error("Transient map key missing")
    }

    var pTransient Patient
    err = json.Unmarshal(transientMap["private_data"], &pTransient)
```

Code snippet 5.15: trasactPatientPrivateData method

```
    if err != nil {
      return shim.Error("Failed to decode JSON of private_data")
    }

    patient := Patient{}
    patient.Id = pTransient.Id
    patient.Name = pTransient.Name
    patient.Surname = pTransient.Surname
    patient.RecordHash = pTransient.Record
    patient.FileURL = pTransient.FileURL
    patient.AuthDoctor = make(map[string]Doctor)
    patient.AuthDoctor = pTransient.AuthDoctor

    patientAsBytes, _ := json.Marshal(patient)
    APIstub.PutPrivateData("private_data",pTransient.Id, patientAsBytes)

    auditLog := AuditLog{}
    auditLog.LogID = patient.ID
    auditLog.UserIdentity = userID
    auditLog.UserRole = "doctor"
    auditLog.ChaincodeName = "patient-ehr"
    auditLog.Channel = "communitychannel"
    auditLog.ChaincodeVersion = "v1.0"
    auditLog.FunctionCall = "trasactPatientPrivateData"
    auditLog.RecordHash = pTransient.RecordHash
    s.AuditLog(APIstub, auditLog)

    return shim.Success(nil)
}
```

Code snippet 5.15: trasactPatientPrivateData method (continued)

```
func (s *SmartContract) getPatientPrivate(APIstub shim.
   ChaincodeStubInterface, args []string) sc.Response {

    //Expecting single argument
    if len(args) != 1 {
      return shim.Error("Incorrect number of arguments.")
    }

    //Get user identity
    userID, err := cid.GetID(APIstub)
    if err != nil {
        return shim.Error("Could not get user identity")
    }

    //Check if user has attribute
    err :=cid.AssertAttributeValue(api,"doctor","true")
    if err != null {
        return shim.Error(authErr)
    }
```

Code snippet 5.16: getPatientPrivateData method

```
    patientAsBytes, _ := APIstub.GetPrivateData("private_data", args[0])
    patient := Patient{}
    json.Unmarshal(patientAsBytes, &patient)

    //Check if doctor is on the list
    val, hasKey := patient.AuthDoctor[userID]
    if hasKey != true {
      return shim.Error("Not authorized to view patient's record")
    }

    return shim.Success(patientAsBytes)
}
```

Code snippet 5.16: getPatientPrivateData method (continued)

5.8.3.2 Results of the experiment

The chaincode was created and deployed successfully to the blockchain network. The *trasactPatientPrivateData()* method executed successfully and the data was not stored on the blockchain network; only the proof of the data was stored on the blockchain network. The actual private data was transacted peer-to-peer between Hospital A and Hospital B. The *getPatientPrivateData()* method successfully retrieved the patient's private data. Hospital C could not retrieve the private data because Hospital C was not a part of the private data collection. The organizations not part of the private data collection cannot gain access to private data and this data is automatically purged after the set period has elapsed. After the private data purge, the information was no longer accessible but proof of the transaction remained on the blockchain after the private data purge.

5.8.3.3 Discussion

The current EHR's infrastructure is built around a centralized architecture that may present a single point of failure. EHRs make use of three models to exchange health records, namely the push, the pull, and the view models. These models generally rely on intermediaries that are often provided by third parties through cloud services. One such intermediary would be the DirectTrust messaging protocol. These models lack a standardized means of generating an audit trail and therefore cannot ensure the integrity of data being transacted. Permissioned blockchain technology provides a semi-decentralized and peer-to-peer transaction mechanism that enables healthcare providers to transact patient's health records without relying on a third party. Smart contracts could enable patients to become owners of their health records. Patients can control access to their health records by maintaining a list of authorized users, effectively creating a document of consent stating who is authorized to interact with the records. Polices can, however, be implemented with smart contracts to overwrite the authorization mechanisms. One event could, for example, enable any doctor to gain access to a patient's health record provided that a doctor declares the patient incapacitated. This

event would be recorded on the blockchain for audit purposes. Refer to Section 5.5 for more information about blockchain-based authorization. Blockchain technology can ensure data integrity from data creation to data use by providing a practically permanent and semi-decentralized audit log. Refer to Section 5.6 for a detailed discussion of a blockchain-based audit log. The possible downside of a blockchain-based approach is that organizations would need to be a part of the blockchain network to interact with the health records. DirectTrust only requires a direct address and an internet connection to transact electronic health records.

5.9 Conclusion

This chapter served to address seconday objective three by testing the technical feasibility of the initial report outlined in Chapter 4. The main objective of this chapter was to evaluate the initial report created in Chapter 4 by conducting experiments. The experimental process followed in this study was outlined in detail. Hyperledger Fabric was selected to represent the features permissioned blockchain technology could provide to EHRs. The Hyperledger Fabric blockchain network setup was also discussed in detail. Various blockchain interventions have been tested through the use of experimentation. The results of the experiments were also outlined in detail.

Experiment 1 focused on the authentication process of EHRs. Certificate-based authentication with blockchain technology provides an improved authentication process. Although certificate-based authentication is an improvement, it does come with a new set of challenges. The private key of a user's digital certificate needs to be stored in a secure location.

Experiment 2 focused on the authorization process of EHRs. Blockchain technology leveraging smart contracts with attribute-based access control (ABAC) provides an improved authorization process. Attribute-based access control provides a more granular and fine-grained access control model than the traditional role-based access control (RBAC). Access to smart contract methods is restricted based on a set of attributes assigned to a user. Smart contracts also enable patients to become the owners of their electronic health records. Consent to access a patient's health records can also be encoded into the smart contract. The outlined authorization process could be viewed as complex and difficult to implement and policies need to be in place to provide doctors with emergency access to a patient's health records.

Experiment 3 focused on the audit log process of EHRs. A blockchain-based audit log provides an improved process compared to the centralized approached followed by traditional EHRs. The permissioned blockchain audit log approach is semi-decentralized and audit logs are stored in an append-only tamper-evident data structure that provides a trusted tamper-evident audit log.

Experiment 4 focused on the data storage process of EHRs. Electronic health records are traditionally stored in a centralized architecture such as a relational database. Permissioned blockchain technology provides a semi-decentralized storage mechanism that stores data in an append-only, tamper-evident structure. The permissioned blockchain approach could therefore provide a higher degree of data integrity. The POPI act dictates that a patient may request that their personally identified information (PII) is purged from a service. This policy cannot be satisfied by blockchain technology, as the data is stored in a rather permanent fashion. The workaround for this would be to store data hashes about a patient's health record.

Experiment 5 focused on the data transaction process of EHRs. The traditional transaction process followed by EHRs lacks a standardized audit log process and handles patient consent in an ad-hoc fashion. The patient's data is also traditionally transacted with the assistance of a third party service. The actions performed with the permissioned blockchain approach are recorded and this provides a higher degree of data integrity. Patient consent could be encoded into smart contracts to streamline the transaction process. The data is transacted in a peer-to-peer fashion, which negates the need for a third party's involvement. The drawback of the blockchain approach is that organizations are required to be a part of the blockchain network to transact information. This chapter provided evidence of the potential of permissioned blockchain technology pertaining to EHRs.

Chapter 6

Conclusion and future work

'Don't wish it were easier. Wish you were better.'

Jim Rohn

6.1 Introduction

The objective of this chapter is to outline the lessons learnt from conducting this study. The results of this study will be discussed in terms of its research objectives, problem statement, contribution to knowledge, recommendations and future work. The research objectives will be discussed to understand the process that the study followed to satisfy the problem statement. Future work will also be discussed in terms of future research areas related to health records and blockchain technology.

6.2 Research objectives

The primary objective of this study is to draft a technical report on the applicability of aspects of blockchain technology for the secure storage and distribution of electronic health records. This report was devised with the support of three secondary objectives. The first secondary objective was to identify the policies and regulations governing electronic health records in South Africa. The second secondary objective was to determine which aspects of blockchain technologies are most suitable for the secure storage and distribution of information. The third secondary objective was to conduct experiments with various blockchain technologies to determine which aspects of blockchain technology would be suited for the secure storage and distribution of electronic health records.

A literature review was conducted on two topics. The first literature review topic identified the policies and regulations governing electronic health records in South Africa, thereby meeting the first secondary objective, which is important as policies and regulations should be adhered to when handling sensitive information, such as health records.

The first literature review topic revealed that currently there was no specific privacy and data protection statute for electronic health records in South Africa. Therefore, generic privacy and data protection laws were discussed in terms of how they relate to electronic health records. The policies and regulations explored by the first literature review topic were as follows: The South African National Health Act (SANHA), 2003 (Act No 61 of 2003); the Health Professions Council of South Africa (HPCSA); the Electronic Communications and Transactions Act (ECTA), 2002 (Act No 25 of 2002) and the PoPIA, 2013. The first secondary objective was satisfied through the first literature review topic. The second literature review topic served to contrast blockchain technologies suitable for storing and sharing information, thereby meeting the second secondary objective of the study. Various blockchain technologies were contrasted to determine which technologies and aspects of blockchain technology would ideally be suited for the secure storage and distribution of information. The literature from the second topic revealed that permissioned blockchain technologies were well suited to handle sensitive information. The second secondary objective was, therefore, addressed through the second literature review topic.

Inductive reasoning was applied to devise an initial report from the literature review topics, which were discussed previously. Experiments were conducted to determine which aspects of blockchain technologies could be used for the secure storage and distribution of electronic health records, thereby meeting the third secondary objective. The results of the experiments revealed that aspects of permissioned blockchain technologies could be used to enhance traditional EHR infrastructure. Hyperledger Fabric was identified as a permissioned blockchain technology suited for the enhancement of electronic health records. Experiments conducted with Hyperledger Fabric were outlined and discussed in detail. The third secondary objective was, therefore, satisfied through experimentation.

Inductive reasoning was also applied to refine the initial report based on the lesson learnt from the conducted experiments. The outcome of the inductive reasoning was a generalized technical report presented in Appendix A. The technical report outlined the applicability of permissioned blockchain technology as a secure storage and distribution mechanism for electronic health records. The primary objective was, therefore, satisfied by a technical report.

6.3 Problem statement

The problem statement of this study is as follows: Electronic health records are commonly stored and distributed in a way that represents a single point of failure and ownership while having to adhere to specific privacy and storage requirements. The problem statement was satisfied by addressing the primary and secondary objectives of this study. The first and second secondary objectives explored the current state of electronic health records and blockchain technology. The experiments that were conducted, revealed that aspects of blockchain technology could address the challenges that

electronic health records currently face. Experiments were conducted with blockchain technology to address the third secondary objective. The primary objective was to devise a technical report which outlined the applicability of blockchain technology as a secure storage and distribution mechanism for electronic health records. The primary and secondary objectives, in turn, therefore, satisfied the problem statement.

6.4 Thesis statement

The thesis statement of this study is as follows: *Aspects of blockchain technology present a viable alternative for the secure storage and distribution of electronic health records.* The thesis statement is supported by the findings of this study. The technical report serves as a summarized version of how permissioned blockchain technology could be a viable alternative to aspects of the traditional infrastructure employed by electronic health records.

6.5 Contribution to knowledge

The main deliverable of this study was to devise a technical report. The technical report serves as a generalized overview of the applicability of blockchain technology as a secure storage and distribution mechanism for electronic health records. Refer to Appendix A for the developed technical report. The technical report provides an overview of applying permissioned blockchain technology to aspects of traditional electronic health records infrastructure. Blockchain technology is a complex concept to understand and learn. The technical report can assist in conceptualizing the key concepts of permissioned blockchain technology and the promise it holds for the future of EHRs. This technical report can, therefore, be seen as a starting point for researchers interested in researching permissioned blockchain technology for EHRs.

6.6 Future Work

The focus of this study was to identify the benefits of blockchain technology for electronic health records. The following areas could be relevant for future research:

Area 1: Future research could be conducted in developing a framework based on permissioned blockchain technology for electronic health records. Developing a framework could be thought of as the next logical step for building on the findings of this study. The majority of permissioned blockchain frameworks are tailored towards general-purpose applications. There is a requirement for developing a blockchain framework tailored towards EHRs, specifically to capitalize on the advantages blockchain technology provides while adhering to specific privacy requirements. A framework could also include the process of systematically incorporating blockchain technology into existing EHR systems.

Area 2: Future research could also be conducted into exploring the benefits that the Internet of Things (IoT) and blockchain technology could provide to the evolution of personal health records (PHR). Recent trends suggest that patients should have control over their health records and be involved throughout the health care process. The main goal of PHR is to effectively enhance transparency and privacy by giving patients control over every aspect of their health records. PHRs could enhance the health care process by providing a comprehensive view of a patient's overall health and lifestyle. Smart IoT health tracking could provide real-time information on the status of a patient's health. IoT devices could be developed to track patients' information in real-time, such as their heart rate, blood pressure, temperature, location and more. The security of IoT and PHRs are currently the stumbling block for the realization of such a system. Blockchain technology could address the security and privacy issues related to PHRs and IoT health tracking devices.

6.7 Conclusion

This chapter explored various components related to this study. The problem statement and research objectives were discussed to understand the process the study followed to satisfy the problem statement. The contribution of this study has been outlined in terms of the development of a technical report, which is attached as Appendix A. Future research has also been discussed concerning blockchain technology for the health care industry. This study has taken the researcher through a journey of self-discovery and exploration through the current state of electronic health records and blockchain technology. The insights gained through the research process have made the researcher aware of the importance of privacy and security with regards to information systems. Sharing electronic health records across the health care industry could be seen as an easy goal to achieve. However, sharing EHRs in a secure and privacy-preserving manner is an extremely intricate process. The researcher strived to explore the feasibility of blockchain technology in the pursuit of a secure, privacy-preserving and unified EHR system. In this pursuit, the researcher identified permissioned blockchain technology as a suitable candidate for further exploration to enhance EHR systems. Experiments with blockchain technology were conducted to test the researcher's theory. The insights gained from these experiments have been discussed in this study. A technical report was also devised to provide an abstract view of the researcher's findings. Consequently, this study has concluded that permissioned blockchain technology could enhance various aspects of the traditional electronic health records infrastructure.

References

Anh, D. T. T., Zhang, M., Ooi, B. C., & Chen, G. (2018). Untangling Blockchain: A Data Processing View of Blockchain Systems. *IEEE Transactions on Knowledge and Data Engineering*, *4347*(c), 1–20. doi: 10.1109/TKDE.2017.2781227

Baliga, A. (2017). *Understanding Blockchain Consensus Models* (Tech. Rep.). doi: Piis0925-3467(02)00124-6\r10.1016/s0925-3467(02)00124-6

Bashir, I. (2017). *Mastering Blockchain* (L. Subramanian, Ed.).

Bergquist, J. H. (2017). Blockchain Technology and Smart Contracts Privacy-preserving Tools. In (p. 45). Retrieved 2019-06-28, from `https://uu.diva-portal.org/smash/get/diva2:1107612/FULLTEXT01.pdf`

Bokefode, J., Ubale, S., & Modani, D. (2014). Analysis of DAC MAC RBAC Access Control based Models for Security. *International Journal of Computer Applications*, *104*(5), 6–13.

Brasser, F., Müller, U., Dmitrienko, A., Kostiainen, K., Capkun, S., & Sadeghi, A.-R. (2017, August). Software grand exposure: SGX cache attacks are practical. In *11th USENIX workshop on offensive technologies (WOOT 17)*. Vancouver, BC: USENIX Association. Retrieved from `https://www.usenix.org/conference/woot17/workshop-program/presentation/brasser`

Brown, R. G., Carlyle, J., Grigg, I., & Hearn, M. (2016). *Corda: An Introduction* (Tech. Rep.). doi: 10.13140/RG.2.2.30487.37284

Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., & Maxwell, G. (2017). Bulletproofs: Efficient Range Proofs for Confidential Transactions. *Cryptology ePrint Archive*. doi: 10.1109/SP.2018.00020

Buys, M., Chb, M. B., Sa, D. A., Anaes, M., & Sa, F. C. A. (2017). IN PRACTICE Protecting personal information : Implications of the Protection of Personal Information (POPI) Act for healthcare professionals. *South African Medical Journal*, *107*(11), 954–956. doi: 10.7196/SAMJ.2017.v107i11.12542

Cachin, C., & Vukolić, M. (2017). Blockchain Consensus Protocols in the Wild.. doi: 10.4230/LIPIcs.DISC.2017.1

Catalini, C. (2017). The Potential for Blockchain to Transform Electronic Health Records. *Harvard Business Review*, 1–7.

Chen, L., Xu, L., Shah, N., Gao, Z., Lu, Y., & Shi, W. (2017). On security analysis of proof-of-elapsed-time (PoET). In *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)* (Vol. 10616 LNCS, pp. 282–297). doi: 10.1007/978-3-319-69084-1_19

Cilliers, L. (2017). Exploring Information Assurance to support Electronic Health Record Systems. In (pp. 1–8).

Coleman, J., Horne, L., & L4, L. X. (2018). *Counterfactual: Generalized State Channels.* Retrieved 2019-07-11, from `https://l4.ventures/papers/statechannels.pdf`

Collider, S. H. (2019). *How secure is my password.* Retrieved 2019-06-28, from `https://howsecureismypassword.net`

Corporation, I. (2017). IBM blockchain foundation developer.. Retrieved 2018-02-20, from `https://www.coursera.org/learn/ibm-blockchain-essentials-for-developers`

Dagher, G. G., Mohler, J., Milojkovic, M., & Marella, P. B. (2018). Ancile: Privacy-preserving framework for access control and interoperability of electronic health records using blockchain technology. *Sustainable Cities and Society*, *39*(August 2017), 283–297. doi: 10.1016/j.scs.2018.02.014

Dekker, M. A. C., & Etalle, S. (2007). Audit-Based Access Control for. *Electronic Notes in Theoretical Computer Science*, *168*(1), 221–236. doi: 10.1016/j.entcs.2006.08.028

Dfinity Technology Overview Series, Consensus System (Tech. Rep.). (2018). Dfinity. Retrieved 2019-02-13, from `internal-pdf://190.248.180.245/dfinity-consensus.pdf`

Ekblaw, A., Azaria, A., Halamka, J. D., & Lippman, A. (2016). *(MedRec) A Case Study for Blockchain in Healthcare: "MedRec" prototype for electronic health records and medical research data* (Tech. Rep.). Retrieved 2019-05-15, from `https://www.healthit.gov/sites/default/files/5-56-onc_blockchainchallenge_mitwhitepaper.pdf`

Emmadi, N., Vigneswaran, R., Kanchanapalli, S., Maddali, L., & Narumanchi, H. (2019). Practical Deployability of Permissioned Blockchains Practical Deployability of Permissioned Blockchains. Springer International Publishing. doi: 10.1007/978-3-030-04849-5

Fink, A. (2010). *Conducting research literature reviews: From the internet to paper.* SAGE Publications. Retrieved 2018-11-09, from `https://books.google.co.za/books?id=2bKI6405TXwC`

Franqueira, V. N. L., & Consulting, V. F. I. (2012). in Retrospect. *Computer (New York)*, *45*(6), 81–88. doi: 10.1109/MC.2012.38

Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., & Waters, B. (2013). Candidate indistinguishability obfuscation and functional encryption for all circuits (extended abstract). *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, 40–49. doi: 10.1109/FOCS.2013.13

Gmbh, B. (2018). *The Blockchain Database* (Tech. Rep. No. May). BigchainDB. Retrieved 2019-03-21, from `https://www.bigchaindb.com/whitepaper/`

Golafshani, N. (2003). Understanding Reliability and Validity in Qualitative Research. *The Qualitative Report*, *8*(4), 597–607. doi: 10.3367/UFNr.0180.201012c.1305

Goodman, L. (2016). *Tezos - White paper* (Tech. Rep. No. July). Tezos. Retrieved 2019-07-13, from `https://tezos.com/static/papers/white_paper.pdf` doi: 10.5663/aps.v1i1.10138

Greenspan, G. (2013). *MultiChain Private Blockchain* (Tech. Rep.). Retrieved 2018-10-02, from http://www.multichain.com/download/MultiChain-White-Paper.pdf doi: 10.1053/sonc.2002.32894

Griggs, K. N., Ossipova, O., Kohlios, C. P., Baccarini, A. N., Howson, E. A., & Hayajneh, T. (2018, Jun 06). Healthcare blockchain system using smart contracts for secure automated remote patient monitoring. *Journal of Medical Systems*, *42*(7), 130. doi: 10.1007/s10916-018-0982-x

Health Professions Council of South Africa. (2014). *HPCSA Booklet 10: General ethical guidelines for good practice in telemedicine.*

Health Professions Council of South Africa. (2016a). *HPCSA Booklet 5: Confidentiality: protecting and providing information.*

Health Professions Council of South Africa. (2016b). *HPCSA Booklet 9: Guidelines on the keeping of patient records.*

Hiraman, B. R. (2018). A Study of Apache Kafka in Big Data Stream Processing. In *2018 international conference on information , communication, engineering and technology (icicet)* (pp. 1–3). IEEE.

Hutton, J. (2016). *Pediatric biomedical informatics: Computer applications in pediatric research.* Springer Singapore.

Hyperledger. (2017). Hyperledger Composer. *Hyperledger Composer Overview*, 1–2. Retrieved 2019-06-16, from https://hyperledger.github.io/composer/latest/introduction/introduction.html

Hyperledger. (2019). Hyperledger Fabric Documentation Release master [Computer software manual]. Retrieved 2019-02-29, from https://readthedocs.org/projects/hyperledger-fabric/downloads/pdf/master/

Hyperledger Architecture Working Group. (2017). Hyperledger Architecture. In (Vol. 1, pp. 1–15). Retrieved 2019-01-11, from https://www.hyperledger.org/wp-content/uploads/2017/08/HyperLedger_Arch_WG_Paper_1_Consensus.pdf

J.P. Morgan Chase. (2018). *Quorum Whitepaper* (Tech. Rep.). Retrieved 2018-12-20, from https://github.com/jpmorganchase/quorum/blob/master/docs/QuorumWhitepaperv0.2.pdf

Katurura, M., & Cilliers, L. (2016). The extent to which the POPI act makes provision for patient privacy in mobile personal health record systems. *2016 IST-Africa Week Conference*, 1–8. doi: 10.1109/ISTAFRICA.2016.7530595

Kedar Iyer, Rene Madsen, Solomon Lederer, Michael Wuehler, Joseph J. Bambara, P. R. A. (2018). *Solutions* (illustrate ed.). McGraw-Hill Education.

Kothari, C. R. (2004). *Research Methodology: Methods & Techniques.* New Age International (P) Ltd., Publishers. doi: 10.1017/CBO9781107415324.004

Kshetri, N. (2017). Blockchain ' s roles in strengthening cybersecurity and protecting privacy. *Telecommunications Policy*, *41*(10), 1027–1038. doi: 10.1016/j.telpol.2017.09.003

Kshetri, N. (2018). Blockchain and Electronic Healthcare Records. *IEEE Computer Society*, *51(12)*(December), 59–63. doi: 10.1109

Kumar, M., & Singh, C. (2017). *Building Data Streaming Applications with Apache Kafka.* Packt Publishing.

Laurence, T. (2017). *Blockchain For Dummies*. Wiley.

Leibovici, T. (2015). Taking back control of HPC file systems with Robinhood Policy Engine.. doi: 10.1145/1529974.1529978

Liang, X., Zhao, J., Shetty, S., Liu, J., & Li, D. (2017). Integrating Blockchain for Data Sharing and Collaboration in Mobile Healthcare Applications.. doi: 10.1109/PIMRC.2017.8292361

Lindell, Y. (2018). The Security of Intel SGX for Key Protection and Data Privacy Applications. In (pp. 1–12). Retrieved 2018-08-19, from `https://cdn2.hubspot.net/hubfs/1761386/security-of-intelsgx-key-protection-data-privacy-apps.pdf`

Mazieres, D., & Mazières, D. (2015). The stellar consensus protocol: A federated model for internet-level consensus. *Stellar Development Foundation*, 1–45. doi: 10.1021/ja982417z

McKendrick, R., & Gallagher, S. (2017). *Mastering docker*. Packt Publishing.

Menachemi, N., & Collum, T. H. (2011). Benefits and drawbacks of electronic health record systems. *Risk Management and Healthcare Policy*, *4*, 47–55. doi: 10.2147/RMHP.S12985

Mosakheil, J. H. (2018). Security Threats Classification in Blockchains..

Orebaugh, A., Ramirez, G., & Beale, J. (2006). *Wireshark & ethereal network protocol analyzer toolkit*. Elsevier Science.

Privacy-preserving blockchain-based electric vehicle charging with dynamic tariff decisions. (2018). *Computer Science - Research and Development*, *33*(1-2), 71–79. doi: 10.1007/s00450-017-0348-5

Pussewalage, H. S. G., & Oleshchuk, V. A. (2016). An Attribute Based Access Control Scheme for Secure Sharing of Electronic Health Records. In *2016 ieee 18th international conference on e-health networking, applications and services (healthcom)* (pp. 1–6). IEEE. doi: 10.1109/HealthCom.2016.7749516

A review of cross organizational healthcare data sharing. (2015). *Procedia Computer Science*, *63*(1), 425–432. Retrieved from `http://www.colleaga.org/sites/default/files/12-55-blockchain-based-approach-final.pdf` doi: 10.1016/j.procs.2015.08.363

Ronquillo, J. G., Winterholler, J. E., Cwikla, K., & Szymanski, R. (2018). Brief Communication Health IT , hacking , and cybersecurity : national trends in data breaches of protected health information. In (Vol. 1, pp. 15–19). doi: 10.1093/jamiaopen/ooy019

Saraf, C., & Sabadra, S. (2018). Blockchain platforms: A compendium. *2018 IEEE International Conference on Innovative Research and Development, ICIRD 2018* (May), 1–6. doi: 10.1109/ICIRD.2018.8376323

Sauce, B., & Matzel, L. D. (2017). *Inductive Reasoning*. Springer International Publishing AG 2017. doi: 10.1007/978-3-319-47829-6

Schwartz, D., Youngs, N., & Britto, A. (2014). *The Ripple protocol consensus algorithm* (Tech. Rep.). Retrieved 2019-05-25, from `http://www.naation.com/ripple-consensus-whitepaper.pdf`

Seol, K., Kim, Y.-g., Lee, E., Seo, Y.-d., & Baik, D.-k. (2018). Privacy-Preserving

Attribute-Based Access Control Model for XML-Based Electronic Health Record System. In *Ieee access* (Vol. 6, pp. 9114–9128). IEEE. doi: 10.1109/ACCESS.2018.2800288

Sharpe, C. (1999). *Medical records review and analysis.* Auburn House.

Sherman, A. T., Javani, F., Zhang, H., Golaszewski, E., & County, B. (2019). On the Origins and Variations of Blockchain Technologies. In *Copublished by the ieee computer and reliability societies.* IEEE. doi: 10.1109/MSEC.2019.2893730

Siim, J. (2017). Proof-of-Stake Research Seminar in Cryptography. In (pp. 1–9). Retrieved from `https://i.blackhat.com/us-18/Wed-August-8/us-18-Narula-Heilman-Cryptanalysis-of-Curl-P-wp.pdf`

Song, X., & Wang, Y. (2017). Homomorphic Cloud Computing Scheme Based on Hybrid Homomorphic Encryption. In (pp. 2450–2453).

South Africa. (2002). *Electronic Communications and Transactions Act* (Vol. 446).

South Africa. (2003). *National Health Act No.61 of 2003.*

South Africa. (2013). *Protection of Personal Information Act No. 4 of 2013.*

Swartz, P. (2017). Personal information and regulatory requirements for direct marketing: A South African insurance industry experiment. *South African Institute of Electrical Engineers*, *108*(June), 56–70.

Tendermint. (2018). *Tendermint Documentation* (Tech. Rep.). Retrieved 2018-06-28, from `https://media.readthedocs.org/pdf/tendermint/master/tendermint.pdf`

Thakkar, M., & Davis, D. C. (2006). Risks, barriers, and benefits of EHR systems: a comparative study based on size of hospital. *Perspectives in health information management / AHIMA*, *3*, 5.

Thakur, M. (2017). *Authentication , Authorization and Accounting with Ethereum Blockchain Department of Computer Science* (Unpublished doctoral dissertation). UNIVERSITY OF HELSINKI.

Townsend, B. (2017). *Privacy and Data Protection in Health Africa* (Unpublished doctoral dissertation). University of Cape Town.

Tuyikeze, T., & Pottas, D. (2010). Information Security Management and Regulatory Compliance in the South African. *Computer Science and Information systems*, *6*(4), 754–761.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A. (2012). *Experiment Process* (Vol. Springer-V). doi: 10.1007/978-3-642-29044-2

Ziglari, H. (2017). Security in EHR System. In *Evaluating cloud deployment models based on security in ehr system.*

Appendix A

Technical Report

A.1 Introduction and objective

There has been a rise in the adoption of electronic health records (EHRs) over recent years (Hutton, 2016). An electronic health record can be seen as an electronic version of a patient's medical history and carry potential benefits, such as improving the quality of care, reducing medical errors, reducing costs, saving time and enhancing the availability of medical records via electronic means (Thakkar & Davis, 2006). However, EHR systems can encounter several challenges in the form of data breaches, privacy compromises, interoperability, audibility, and fraud. EHR systems currently utilize a centralized architecture that requires a centralized authority of trust and leaves medical records vulnerable due to a single point of failure (Liang et al., 2017). Electronic health records deal with highly sensitive information, such as a patient's demographics, diagnoses, vital signs, past medical history, etc. (Menachemi & Collum, 2011). Cybercriminals view EHRs as lucrative targets and as a result, EHRs are constantly vulnerable to cyberattacks. The number and severity of successful cyberattacks on electronic health records are rising (Ronquillo et al., 2018). The conventional model that is currently utilized by electronic health record systems can no longer ensure the security and privacy of a patient's health records (Kshetri, 2018). Desirable features of blockchain technology such as decentralization, immutability, audibility, and transparency could be used to enhance the security and privacy of EHRs (Emmadi et al., 2019). This report serves to address the primary objective which is to compile a technical report on the applicability of aspects of blockchain technology to the secure storage and distribution of electronic health records.

A.2 Focus of the report

This report aimed to evaluate the current state of electronic health records and blockchain technology, as aspects of blockchain technology could improve the traditional EHR infrastructure. This report also focused on how particular aspects of blockchain technology could be utilized to improve the current state of electronic health records.

A.2.1 Electronic health records EHR

The healthcare industry has been evolving over recent years. Enhancements in information technology have fuelled the evolution from paper-based records to electronic health records (Seol et al., 2018). Healthcare providers frequently store electronic health record in a centralized architecture such as a relational database (Griggs et al., 2018). EHR systems are commonly built with a cloud computing infrastructure to facilitate the storing and sharing of health records (Ziglari, 2017). Cloud computing can be seen as a service that provides clients access to remote computing resources over the internet. Computing resources can range from servers, storage, databases, analytics, etc. Cloud computing provides benefits such as availability, flexibility, scalability, and reduced operational cost (Ziglari, 2017). Sharing EHRs with a cloud computing infrastructure is efficient and effective. Thus, the cloud computing infrastructure works well for EHRs from an availability standpoint. However, the increased accessibility has brought forth challenges like security and privacy issues. Cyber-criminals view EHRs as lucrative targets since they contain vast amounts of sensitive information (Griggs et al., 2018). Therefore, EHRs face constant cyber-attacks. Thus, aspects of the conventional EHR model should be refined. An overview of the current EHR model is illustrated in Figure A.1.

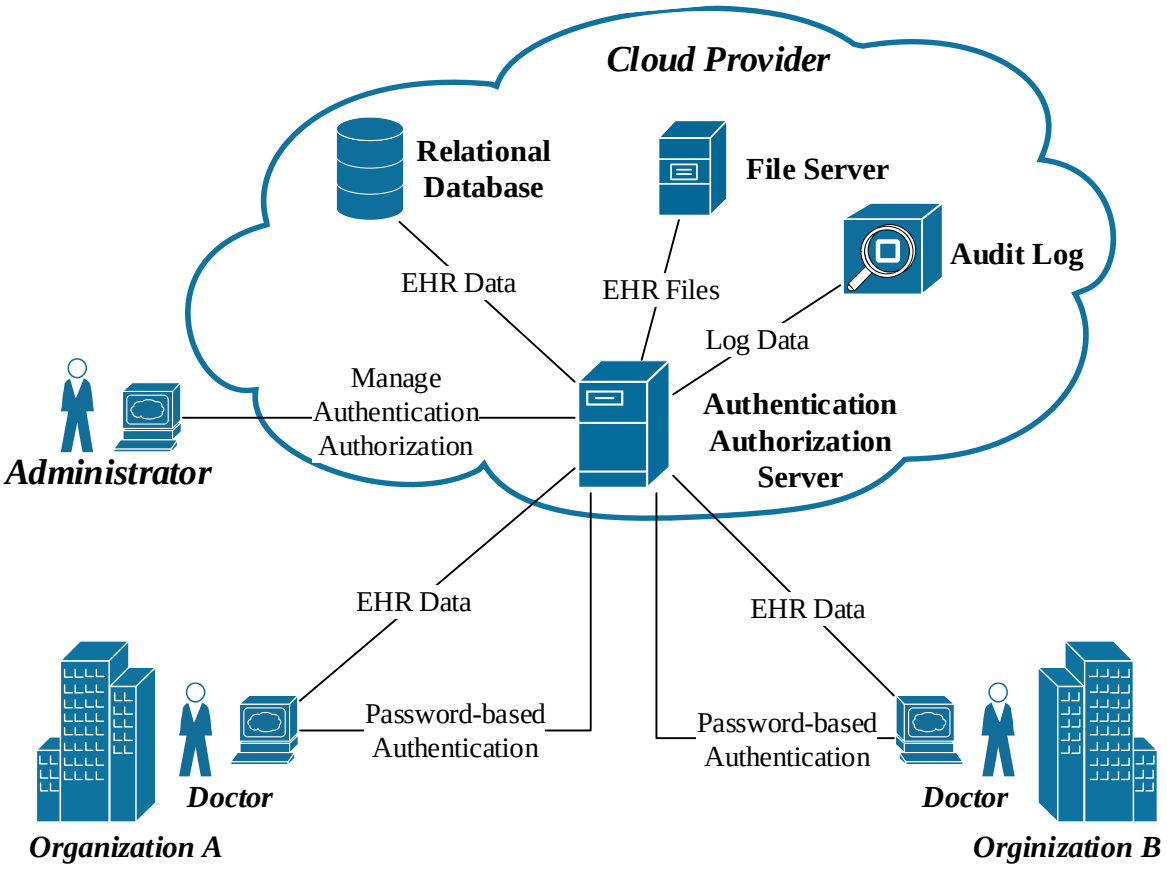

Figure A.1: EHR system overview with blockchain intervention diagram

Recent reports illustrate that over 112 million electronic health records have been exposed through data breaches in 2015 (Kshetri, 2018). The result of these data breaches suggests that the centralized infrastructures used by EHRs do not work from a privacy and security standpoint.

A.2.2 Blockchain technology

Blockchain technology is a type of distributed ledger technology that differs from other distributed ledger technology by bundling unrelated data into blocks that are chained together in a linked-list manner, hence the name blockchain (Bashir, 2017, p. 18). In simple terms, blockchain could be seen as a distributed database controlled by a group of individuals. Adding records to the database requires users to propose a transaction, which is then broadcast to a peer-to-peer network consisting of computers known as nodes. The network of nodes validates the transaction and the user's status using known algorithms. The verified transaction is combined with other transactions to form a block, which is then appended to the blockchain in a way that is practically permanent and unalterable (Bashir, 2017, p. 27). The Blockchain innovation is fundamentally built on old technologies that are used in new ways. The terms Bitcoin and blockchain are being used interchangeably. These two terms, however, are not the same. Blockchain technology is the underlying technology used in the Bitcoin protocol to facilitate the secure transfer of Bitcoin (Bashir, 2017, p. 111). The term Bitcoin is the name of the cryptocurrency that powers the Bitcoin network. Blockchain technology is not limited to the application of cryptocurrency (Bashir, 2017, p. 23).

A.2.3 Consensus algorithms in blockchain technology

The concept of consensus is used by blockchain technology to ensure that all the peers in a network agree to a single history of transactions. (Bergquist, 2017). There are mainly two types of consensus algorithms,namely proof-based and Byzantine fault tolerance-based. Proof-based consensus is were a leader is elected based on having some kind of proof that grants them the authority to propose a new value. Byzantine fault tolerance-based consensus is were rounds of votes are used to propose a new value (Bashir, 2017, p. 28). Examples of consensus algorithms include proof-of-work, proof-of-stake, proof-of-elapsed-time, proof-of-authority, practical Byzantine fault tolerance and many more.

A.2.4 Smart contract

Smart contracts were proposed in the late 1990s by Nick Szabo (Bashir, 2017, p. 198). A smart contract in blockchain terms is used to digitally facilitate, enforce and verify that all the terms of a contract (business logic) are met before a transaction can take place (Bergquist, 2017). Smart contracts eliminate the need for a third party's involvement. Once a smart contract has been executed it is tractable and irreversible (Bashir, 2017, p. 198). A smart contract's logic should thus be thoroughly tested before it is deployed in a production blockchain network.

A.2.5 Privacy and confidentiality

Enterprise blockchain networks, also known as permissioned blockchains, are being developed to cater to enterprise use cases (Emmadi et al., 2019). Public blockchain networks lack key features such as confidentiality, privacy, user identity, authorization and audibility, and these features are therefore being incorporated into permissioned blockchain networks to accommodate enterprise use cases. One of the challenges to the realization of enterprise-grade blockchain technology is privacy (Bashir, 2017, 461). Enterprise entities that are part of a consortium network require information to be shared privately and securely. Consortium networks also require a level of isolation to ensure that only authorized parties can gain access to confidential information. The nature of blockchain technology is to promote transparency through the use of a shared ledger. Transparency and privacy are considered to be opposing forces and permissioned blockchain technology thus strives to achieve a balance between transparency and privacy (Emmadi et al., 2019). Research is currently underway to improve the privacy and confidentiality of blockchain technology.

A.2.6 Comparing blockchain technology

Various aspects of blockchain technologies have been explored and are compared in this section. The comparisons are illustrated by means of tables, each of which focuses on various aspects of blockchain technology. Table A.1 compares different types of blockchain networks, namely public, permissioned and private. Public blockchain networks are open to the public and anyone can partake in the consensus process (Bashir, 2017, p. 26). Everyone can view all the transactions in a public blockchain network. As public blockchain networks utilize pseudonymous identities, it is challenging to establish and control the identities of participants. Permissioned blockchain systems are controlled by a quorum of organizations and as a result, are classified as semi-decentralized. The membership of a permissioned blockchain system is strictly controlled and transactions are generally confidential between participants. Private blockchain systems are commonly owned and controlled by a single organization and private blockchain systems are therefore classified as centralized. Table A.2 compares various consensus algorithms in terms of Byzantine fault tolerance, transaction speed, scalability, and finality. Byzantine fault tolerance refers to the detection and prevention of malicious nodes in a blockchain network. Transaction speed is the maximum throughput that can be expected from a consensus algorithm. Scalability refers to the impact the number of peers in a network has on the overall stability and performance of a blockchain network. Finality refers to the process involved in reaching a consensus state in the blockchain network. Table A.3 compares popular blockchain protocols in terms of network type, consensus algorithm, data privacy, smart contract languages, application, and status.

Table A.1: Comparison of blockchain types.

Criteria	**Public**	**Permissioned**	**Private**
Architecture	Decentralized	Semi-decentralized	Centralized
Immutability	Practically tamper-proof	Tamper-evident	Tamper-evident
Transparency	Full transparency	Semi-transparent	Semi-transparent, No trasnparency
Transaction Speed	Slow	Fast	Fast

Table A.2: Comparison of blockchain consensus algorithms.

Consensus	**Byzantine fault tolerance**	**Transactions per second**	**Scalibility**	**Finality**
Proof-of-work	✓	<100	High	Probabilistic
Proof-of-stake	✓	<1000	High	Probabilistic
Proof-of-elapsed-time	✓	#	High	Probabilistic
Tendermint	✓	<=10k	Low	Deterministic
PBFT	✓	<2000	Low	Deterministic
Raft	x	>10k	Low	Deterministic
Kafka-ordering	x	-	Low	Deterministic
Sumeragi-BFT	✓	-	Low	Deterministic
Scalable-BFT	✓	-	Low	Deterministic

(✓) Supported; (x) Not-supported; (-) Unknown; (#) Depenends on implementation

Table A.3: Comparison of popular blockchain protocols.

Platform	**Type**	**Consensus**	**Data Privacy**	**Smart Contract language**	**Application**	**Status**
Bitcoin	Public	Proof-of-work	-	Go, C++	Crypto-currency	Active
Ethereum	Public, Private	Proof-of-work, Proof-of-stake	-	Solidity, Serpent, LLL	Multi-purpose	Active
Parity Ethereum	Private	Proof-of-authority	-	Solidity, Serpent, LLL	Multi-purpose	Active
Quorum	Permissioned	Raft, Istanbul-BFT	ZKP	Solidity	Multi-purpose	Active
Hyperledegr Fabric	Permissioned, Private	Solo, Kafka	TLS, ZKP, Channels	Go, Java	Multi-purpose	Active
Hyperledegr Burrow	Permissioned	Tendermint	-	Solidity	Multi-purpose	Incubation
Hyperledger Sawtooth Lake	Permissioned, Public	Proof-of-elapsed -time	Intel SGX	Go, C++, etc.	Multi-purpose	Active
Hyperledger Iroha	Permissioned	Sumeragi-BFT	Channels	Java	Multi-purpose	Active
Hyperledger Indy	Permissioned	Redundant-BFT	ZKP	-	Decentralized identity	Active
R3 Corda	Permissioned, Private	Raft	ZKP, State object	Kotlin, Java	Financial	Active
Dfinity	Public	Threshold relay	-	Solidity, Serpent, LLL	Multi-purpose	Incubation
Kadena	Private	Scalable-BFT	Onchain encryption	Pact	Multi-purpose	Active

A.3 Proposed network topology

Figure A.2 illustrates a generic version of a permissioned blockchain network. The network topology is based on the Hyperledger Fabric's network. The proposed network consists of multiple components known as clients, certificate authorities (CAs) and peers. Clients are used for interacting with the blockchain network, the certificate authorities are responsible for issuing, validating and revoking digital certificates and peers are used for storing the linked-list of blocks. Permissioned blockchain platforms make use of different consensus algorithms to manage the synchronization of peers. Consensus algorithms provide various characteristics and permissioned blockchain platforms thus enable developers to replace consensus algorithms as required. Organizations are linked together using their respective peers. Each organization requires at least one of each network component, as illustrated in Figure A.2. The organizations are advised to include multiple peers for internal redundancy purposes. The following sections of this report cover the improvements that blockchain technology could provide for the traditional electronic healthcare records.

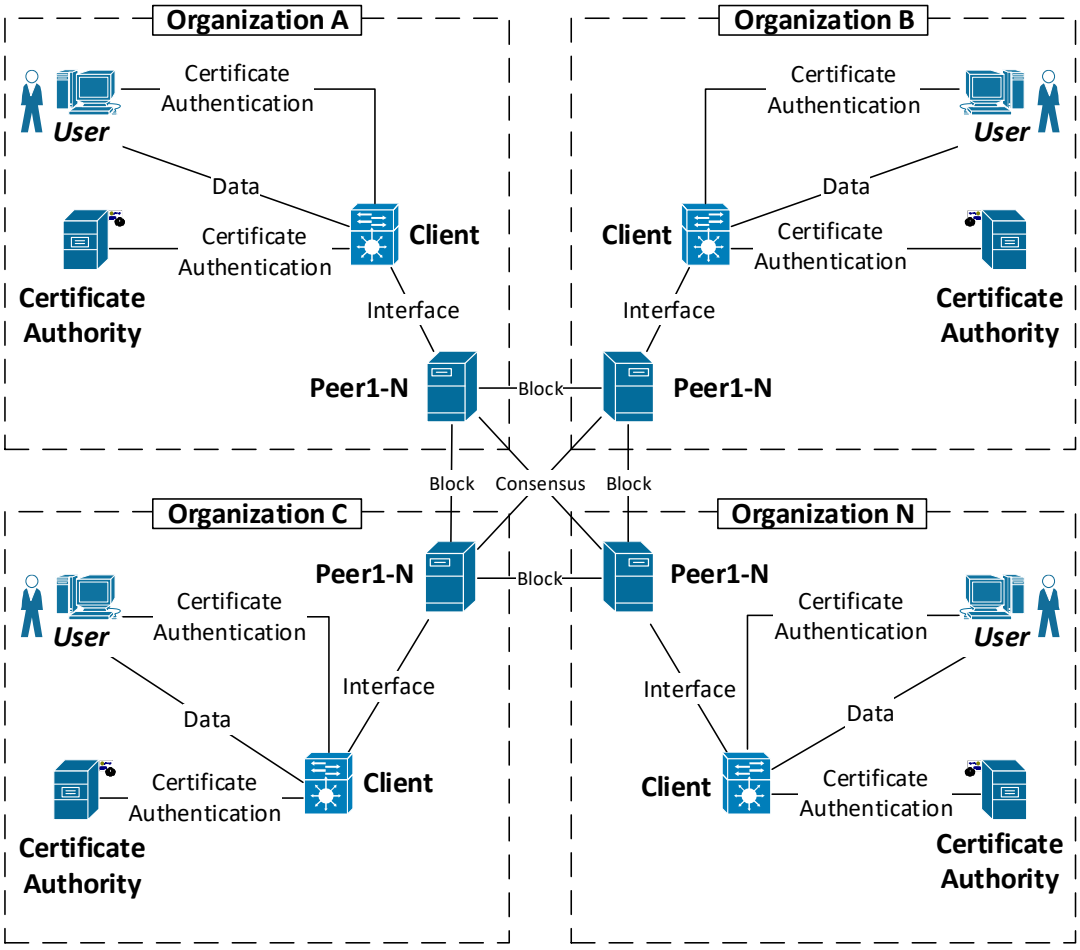

Figure A.2: Proposed network topology

A.4 Blockchain-based authentication

Authentication is used to identify a user requesting access to the system. Systems need to be able to identify users to restrict access to system functions. Users' identities are also required for audit purposes (Cilliers, 2017).

A.4.1 Intent

Enterprise systems traditionally utilize password-based authentication, which often relies on a centralized architecture such as a relational database (Kshetri, 2018). Blockchain technology can leverage smart contracts and public key infrastructure (PKI) to replace password-based authentication with certificate-based authentication. This section covers the implementation of a blockchain-based authentication mechanism for electronic health records (EHRS).

A.4.2 Problem

Electronic health record systems make use of password-based authentication (Kshetri, 2018). Passwords are often created by humans and are therefore often considered to be weak and easily cracked by utilizing techniques such as social engineering, password guessing, and brute-forcing (Kshetri, 2017). Passwords are commonly stored in a centralized relational database, which may represent a single point of failure. When an authentication database is compromised passwords can be stolen, which can lead to data breaches that could lead to several additional data breaches. Password-based authentication used in a centralized architecture is thus considered vulnerable to cyberattacks (Mosakheil, 2018).

A.4.3 Solution

Permissioned blockchain technology can utilize smart contracts and certificate-based authentication to replace the traditional password-based authentication mechanism. Certificate-based authentication removes the human factor from the authentication process. Certificates are often created with a 2048bit key size, which is much larger than an average password size. It is considered to be impractical to brute force a certificate, as it would take a standard desktop computer years to crack. Certificates come with an expiration date witch can reduce the risk of prolonged data exposure. Blockchain technology can leverage smart contracts to validate user certificates and effectively mitigate the risk of a single point of failure.

A.4.4 Structure

The information flow of the blockchain-based authentication model is illustrated in Figure A.3. Administrators of the EHR system can create and revoke digital certificates and the certificates are issued and revoked by the certificate authority. When a user's digital

certificate has been revoked, a certificate revocation list (CRL) is generated. The peers in the blockchain network receive a CRL update command to update the CRL stored on the peers. Users can authenticate with the EHR system by providing the system with their certificate. The certificate is then passed to the client, which is then sent to a peer in the network. The peer authenticates the user by running the authentication smart contract, which validates the certificate and returns a response. This response determines the authentication status of a user.

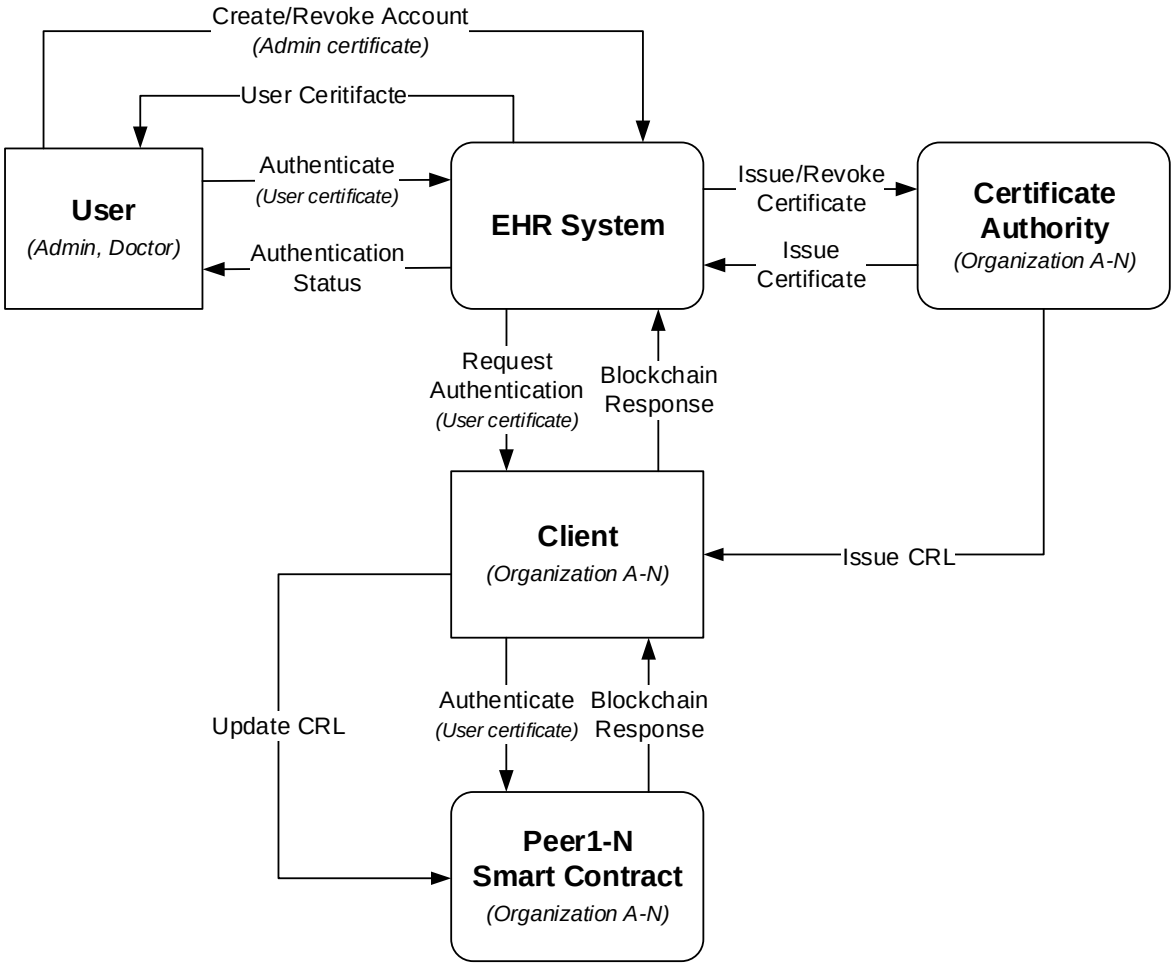

Figure A.3: Blockchain authentication data flow diagram

A.4.5 Pseudocode

The following section outlines a generalized authentication smart contract function based on experiments conducted with Hyperledger Fabric. The pseudocode outlined in Code snippet A.1 serves as an example of an authentication function that is based on experiments conducted with Hyperledger Fabric. The function described in this section

can be implemented with any permissioned blockchain technology that supports smart contracts and the integration of certificate authorities.

```
Function AuthenticateUser(Certificate user_certificate ) {

    Do sanitization check on the function inputs;
    Get the users certificate;
    Validate users certificate;
    if the user certificate has been signed by an trusted CA then

        if user certificate is not present on CRL then
            Grant user access;
        else
            Deny user access;
        end if

    else
        Deny user access;
    end if

}
```

Code snippet A.1: Authentication pseudocode

A.4.6 Applicability

Use certificate-based authentication when it is necessary to remove the human factor from the authentication process.

History dictates that password-based authentication is not secure enough for sensitive information. The human factor in password-based authentication is the main weak point and as passwords are often created by humans, they are usually short and weak. This is because it is difficult for humans to remember long and complex passwords.

Certificate-based authentication appears to be more secure than password-based authentication. Digital certificates are created by computers that understand the importance of entropy. In 2019 certificates are created with a key size of 2048bits, which translates to 256 characters. This would take a standard computer a quadrillion years to crack.

Use certificate-based authentication in conjunction with blockchain technology to mitigate the risk of a single point of failure.

Passwords are commonly stored in a centralized database. When an authentication database is compromised passwords can be stolen and this can result in breaches that could lead to many more data breaches.

Digital certificates come with two keys known as a private key and a public key. The public key is derived from the private key and both these keys are required for authentication. The private keys of certificates are commonly stored on the user's machine and users are encouraged to safeguard their private keys by storing them in a hardware security module (HSM) or trusted platform module (TPM). Not all certificates are stored in a centralized architecture. The certificate revocation list (CRL) is also distributed across all of the peers in the blockchain network. It is thus increasingly difficult to tamper with the authentication mechanism, as the majority of the peers in the blockchain network need to be compromised.

A.5 Blockchain-based authorization

Authorization is used to determine the actions an authenticated user can perform on a system (Cilliers, 2017). System functions should be restricted with the use of appropriate authorization mechanisms and only users that need access to functions should be able to gain access. Restricting rights in this manner mitigate the risk of unauthorized disclosure of information (Seol et al., 2018).

A.5.1 Intent

Enterprise systems predominantly utilize centralized authorization architecture. Blockchain technology can replace the prominent centralized authorization architecture with a distributed architecture. Enterprise systems commonly rely on a role-based access control (RBAC) model to restrict access to information. Blockchain technology can leverage public key infrastructure (PKI) and smart contracts to create a distributed attribute-based access control (ABAC) model. This section discusses the implementation of a blockchain-based authorization mechanism for electronic health records (EHRs).

A.5.2 Problem

Electronic health record (EHR) systems commonly make use of a role-based access control model to enforce authorization (Seol et al., 2018). The role-based access control rules are commonly encoded into the application code that is hosted on a centralized server and when a user requests access to resources the user requires a specific role. The user's role is generally stored and mapped in a centralized relational database and this may present a single point of failure. If the relational database has been compromised, an attacker could add or remove privileges or hijack the highest privileged user account, resulting in unauthorized access to a system. The role-based access control model does not deal well with complex attributes such as object attributes, subject attributes, and contextual attributes. The role-based access control model can only restrict access to specific actions based on the user's role. Each customized user role would require a new role, resulting in role explosion, which could be difficult to maintain. For example, the role of 'doctor' should only be able to access their patients' records and not have access

to all patient records. Supporting this type of constraint with RBAC would require each doctor to have an individual role with access rights to their respective patients (Franqueira & Consulting, 2012). EHRs of today require a fine-grained and dynamic access control mechanism to deal with complex attributes (Seol et al., 2018).

A.5.3 Solution

Permissioned blockchain technology and attribute-based access control (ABAC) to the rescue. Persmssioned blockchain technology makes use of certificated based authentication and the certificates used to authenticate users can also be used to restrict access to resources based on complex attributes. These attributes can be encoded into the user's certificate. Attribute-based access control (ABAC) enables organizations to implement a granular and fine-grained access control model. Access can effectively be restricted based on a set of four attribute types, commonly known as subject attributes, action attributes, object attributes, and contextual attributes. Subject attributes are those related to the user requesting access to the system, e.g., department, role, age, etc. Action attributes describe the actions a user is allowed to perform on a system, e.g., write, read, delete, etc. Object attributes are used to define which object types a user is authorized to access, e.g., department, bank account, medical record, etc and contextual attributes deal with the dynamic aspects of access control, such as the user's location, the time of day, etc. Combinations of these attribute types can support complex and fine-grained access control rules that can be coded into smart contracts to restrict access to information. Smart contracts are distributed across the blockchain network. This solution provides a distributed architecture to combat a single point of failure as well as enhancing the authorization capabilities of the system.

A.5.4 Structure

The information flow of the blockchain-based authorization model is illustrated in Figure A.4. The data flow diagram is based on the authentication data flow diagram. The authorization data follow the diagram, therefore, omits the authentication process. Users should first be authenticated before the authorization process is performed. This structure assumes a user has a valid certificate when attempting to execute an operation on the EHR system. The EHR system contacts the blockchain client to invoke the authorization smart contract stored on the peers in the network. The smart contract validates the user attributes stored in the certificate against the authorization rules embedded in the smart contract. The smart contract then returns an authorization response.

A.5.5 Pseudocode

The ensuing section outlines a generalized authorization smart contract function based on experiments conducted with Hyperledger Fabric. The pseudocode outlined in Code snippet A.2 serves as an example of an authorization function that is based on experiments conducted with Hyperledger Fabric. The functionality described in this section

can be implemented on any permissioned blockchain technology that supports smart contracts and certificate-based authentication.

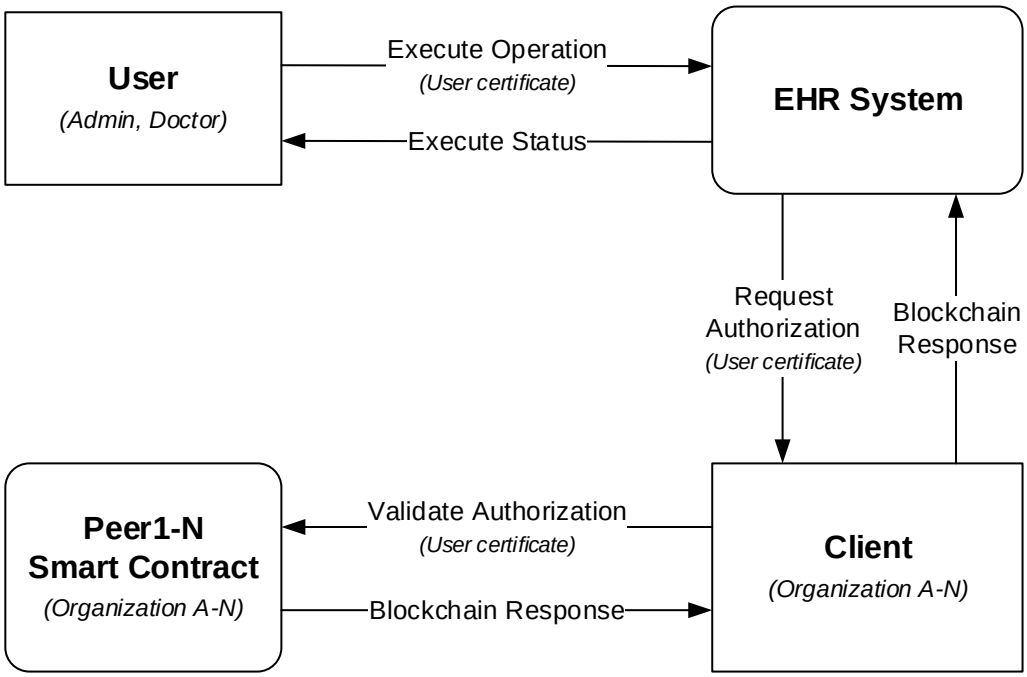

Figure A.4: Blockchain authorization data flow diagram

```
Function RequestAccessToProtectedResources() {

   Do sanitization check on the function inputs;
   Get the users certificate;
   Validate users certificate;
   Get user's identity from certificate;
   Get user's attributes from certificate;

   if the user attributes match the authorization rules
       Grant access to the user;
       //Extra rules can also be checked further;
       if userID is present on a stored list
           Grant further access;
       else
           Deny further access;
       end if;

   else
       Deny access to the user;
   end if;

}
```

Code snippet A.2: Authorization pseudocode

A.5.6 Applicability

Use the Attribute-based access control (ABAC) model when there is a need for a complex and fine-grained authorization model.

Role-based access control (RBAC) can only restrict access to specific actions based on the user's role. Each customized user role would require a new role. Resulting in role-explosion which could be very difficult to maintain. Therefore, the RBAC model is not suitable for restricting access to individual operations or specific data objects.

Attribute-based access control (ABAC) enables organizations to implement a granular and fine-grained access control model. Access can effectively be restricted based on a set of four attribute types that are commonly known as subject attributes, action attributes, object attributes, and contextual attributes. Combinations of these attribute types can support complex and fine-grained access control rules.

Use the attribute-based access control (ABAC) model in conjunction with blockchain technology to provide a distributed access control model.

The role-based access control rules are commonly encoded into the application code that is hosted on a centralized server. When a user requests access to resources that the user requires a specific role, which is generally stored and mapped in a centralized relational database. This may present a single point of failure. If the relational database has been compromised, an attacker could add or remove privileges or hijack the highest privileged user account, resulting in unauthorized access to the system.

The ABAC attributes are encoded into digital certificates that are stored on the user's machine and the access control rules are encoded into smart contracts that are distributed across a network of peers. An attacker would need to either steal an administrator's certificate or compromise the majority of the peers in the blockchain network. These two actions are considered to be impractical, as attacks such as these could be detected and traced because each action on the blockchain network is recorded.

A.6 Blockchain-based audit log

An audit log is a recording of all the actions a user has performed on a system. Audit logs are useful in identifying how, when, where, why and by whom data was accessed, modified and/or leaked. Tampering with audit logs frequently occurs to cover a criminal's tracks (Dekker & Etalle, 2007).

A.6.1 Intent

Enterprise systems predominantly utilize a centralized audit log architecture. Audit logs are commonly stored locally in a file or remotely on a relational database but these

methods of storage are not considered to be immutable. Blockchain technology could provide a distributed and practically immutable audit log.

A.6.2 Problem

Electronic health record systems do not generate standardized audit logs. These audit logs are also stored in a centralized architecture which may present a single point of failure. Electronic healthcare records are often subject to tampering for various reasons, which include insurance coverage, criminal offenses, and more (Kshetri, 2018). Data integrity cannot be assured without an immutable audit log and it is therefore important that EHR systems maintain an immutable audit log to identify criminals tampering with healthcare records. Audit logs also have a key role in identifying culprits responsible for data breaches (Kshetri, 2017).

A.6.3 Solution

Permissioned blockchain technology can be used to generate a semi-decentralized, tamper-evident and standardized audit log for electronic health record systems. Permissioned blockchain networks make use of membership service to identify users interacting in the blockchain network (Bashir, 2017, p. 362). The user's identity can be used to record all the actions performed by the user on the blockchain network. Blockchain technology provides a tamper-evident and peer-to-peer means of storage. This storage model can ensure data integrity from data creation to data retrieval.

A.6.4 Structure

The information flow of the blockchain-based audit log model is illustrated in Figure A.5. The data flow diagram is based on the authentication and authorization data flow diagram. The audit log data flow diagram omits the authentication and authorization process to simplify the diagram. The process assumes that a user has been authenticated. When a user attempts to execute an operation on the EHR system an event is triggered, which sends metadata to the client. Metadata could include details such as the actions performed by a user on an object and the result of the performed actions. The metadata is sent from the client to the peers in the blockchain network. At a later stage, authorized auditors can request the audit log from the EHR system. The audit log could also be used by doctors to validate the integrity of EHR data in their possession.

A.6.5 Pseudocode

The following section outlines a generalized audit log smart contract based on experiments conducted with Hyperledger Fabric. Refer to Code snippet A.4 for pseudocode to append an audit log entry to a blockchain network. Code snippet A.3 outlines pseudocode to retrieve an audit log by range. The functions described in this section can

be ported to any permissioned blockchain network that supports smart contracts and certificate-based authentication.

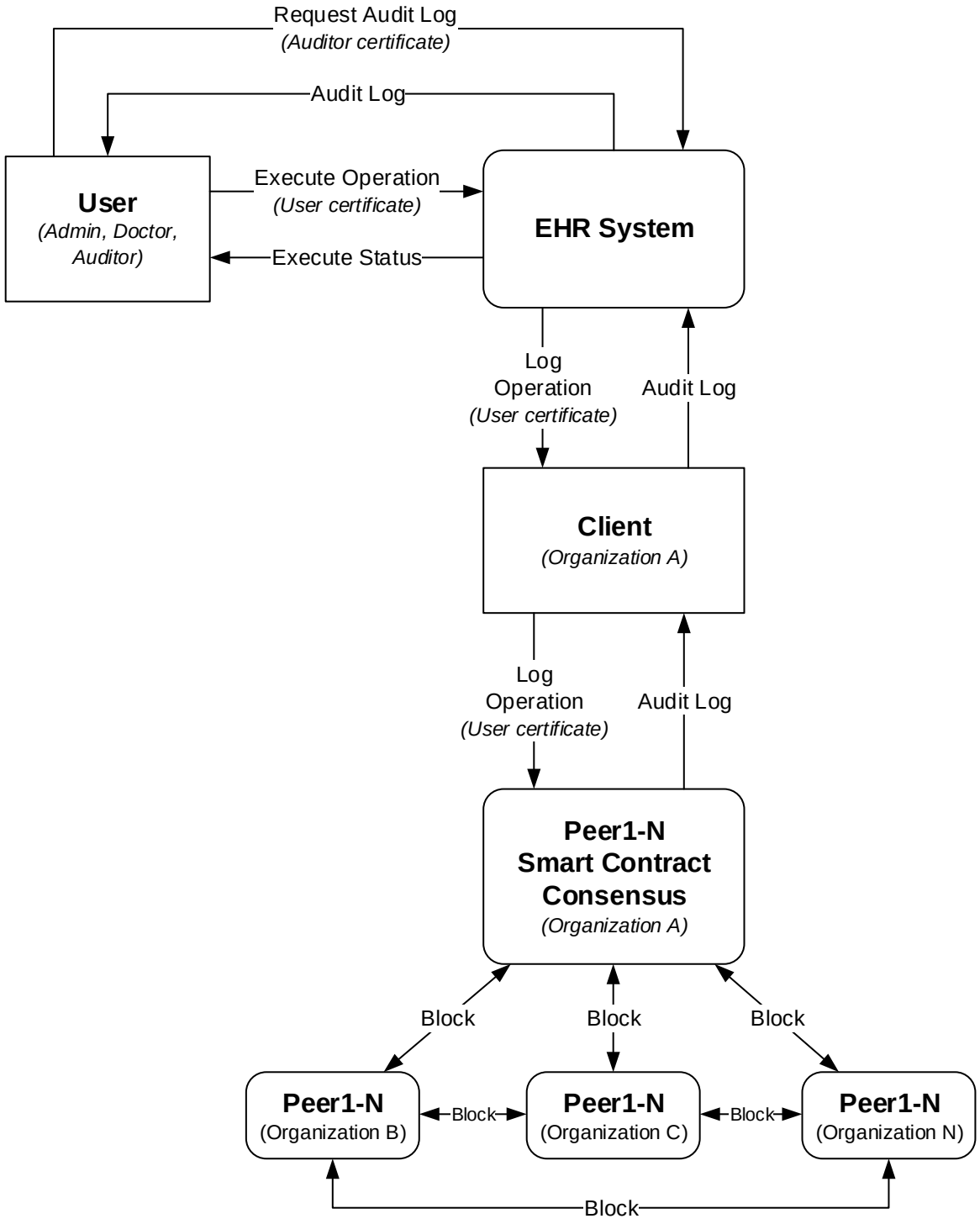

Figure A.5: Blockchain audit log data flow diagram

```
Function AuditLog GetAuditLog(Date startDate, Date endDate) {

   Do sanitization check on the function inputs;
   Get the user certificate;
   Validate user certificate;
   Get user's attributes from certificate;

   if the user attributes match the authorization rules
       return AuditLogByRange(startDate,endDate);
   else
       Deny access to the user;
       CreateUnauthorisedLogEntry(log);
   end if;

}
```

Code snippet A.3: GetAuditLog pseudocode

```
Function AppendAuditLog(AuditLog log) {

   Do sanitization check on the function inputs;
   Get the device certificate;
   if device certificate is not valid then
       Deny access to the device;
   end if

   Get device attributes from certificate;
   if the device attributes match the authorization rules
       Get user certificate;
       if user certificate is valid then
           CreateLogEntry(log);
       else
           CreateUnauthorisedLogEntry(log);
       end if
   else
       CreateUnauthorisedLogEntry(log);
       Deny access to the device;
   end if;

}
```

Code snippet A.4: AppendAauditLog pseudocode

A.6.6 Applicability

Use the blockchain-based audit log model to combat a single point of failure and to ensure the integrity of data stored in a system.

Traditional audit log systems are built around a centralized architecture and audit logs are often stored in a relational database or on a file server. While these methods of storage work practically, they represent a single point of failure. Compromising one of

these storage methods would enable cybercriminals to erase their tracks, thus allowing their actions to remain undetected. As a result, the integrity of the data stored in the relational database cannot be guaranteed.

Blockchain-based audit logs are practically permanent and tamper-evident. Blockchain is an append-only data structure that is distributed across several peers and cybercriminals would have to attack the majority of the peers in the network simultaneously to corrupt the audit log. This attack would not go unnoticed. Even if cybercriminals can hijack a user's account, the changes made by the account would not go undetected. The changes made to the audit log would be appended, leaving the previous records intact. This could then be used to flag suspicious accounts and track the cybercriminals responsible. Data integrity can thus be preserved through the use of blockchain technology.

A.7 Blockchain-based data storage

Data storage is the act of recording information electronically. Data can be stored by utilizing a variety of structures and architectures, all of which have advantages and disadvantages.

A.7.1 Intent

Enterprise systems predominantly utilize a centralized client-server model to store data. Centralized data storage such as a relational database used by enterprise systems provides a high degree of transaction throughput but could be vulnerable due to a single point of failure (Liang et al., 2017). Blockchain technology can replace the common centralized client-server model with a peer-to-peer, semi-decentralized and tamper-evident means of append-only storage for enterprise systems (Bashir, 2017, p. 438).

A.7.2 Problem

Healthcare institutions traditionally make use of a client-server model to maintain their electronic health records, which are generally stored in a centralized architecture, such as a relational database, which represents a single point of failure (Liang et al., 2017). Electronic healthcare records are often subjected to tampering for various reasons, two of which are insurance coverage and criminal offenses. Healthcare providers also often make use of potentially insecure clouds to store shared secrets. The U.S. health insurer, Anthem's, data breach that occurred in December 2014 exposed more than 80 million clients' sensitive information (Kshetri, 2018). This attack is an example of why centralized architectures are often considered easy and lucrative targets.

A.7.3 Solution

Blockchain technology could provide a peer-to-peer, semi-decentralized and tamper-evident means of append-only storage for electronic health record systems (Bashir, 2017, p. 438). This storage model can ensure data integrity from data creation to data retrieval. A single point of failure can also be averted with the use of blockchain technology (Kshetri, 2018). Permissioned blockchain technology can be used to enhance the privacy of data being stored on a blockchain network. Cryptographic techniques such as zero-knowledge proofs can be used to store data privately and ensure that data integrity can be maintained without revealing private information (Bünz et al., 2017). Permissioned blockchain technology can improve the current storage architecture in use by EHR systems.

A.7.4 Structure

The information flow of the blockchain-based storage model is illustrated in Figure A.6. The data flow diagram is based on the authentication and authorization data flow diagrams. The storage data flow diagram omits the authentication and authorization process to simplify the diagram. The process assumes that a user has been authenticated and when that user would like to view, add or edit a patient's record they can do so by contacting the EHR system. The EHR system would then request or send the information to the blockchain client. The blockchain client then forwards the request to one of the peers in the network and the relevant smart contract code is then executed on the peer. The smart contract then proposes a transaction to the blockchain network. This transaction is bundled together into a block and appended to the blockchain. The consensus algorithm ensures that all of the peers are in sync.

A.7.5 Pseudocode

The following section outlines a generalized storage smart contract based on experiments conducted with Hyperledger Fabric. Code snippet A.5 outlines pseudocode for adding or updating a record in the blockchain network. Retrieving records from the blockchain could be achieved with the pseudocode written in Code snippet A.6. The methods explained in this section could be ported to any permissioned blockchain network that supports smart contracts and certificate-based authentication.

```
Function AppendData(Data data) {
   Do sanitization check on the function inputs;
   Get the user certificate;
   Validate user certificate;
   Get user's attributes from certificate;
   if the user attributes match the authorization rules
       CreateDataEntry(data);
```

Code snippet A.5: AppendData pseudocode

```
    else
        Deny access to the user;
    end if;
}
```

Code snippet A.5: continued

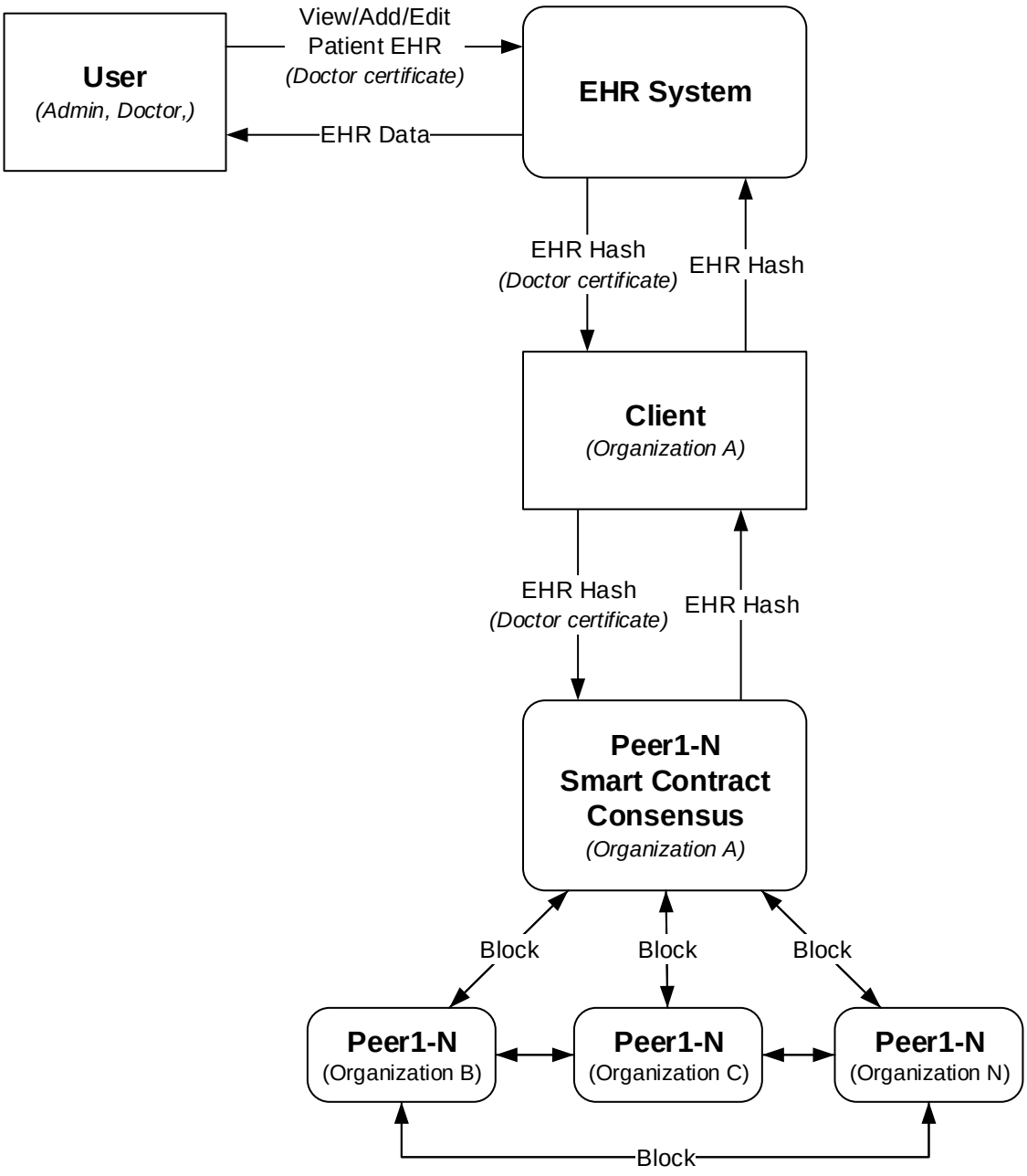

Figure A.6: Blockchain storage data flow diagram

```
response Function GetData(Key key) {

    Do sanitization check on the function inputs;
    Get the user certificate;
    Validate user certificate;

    Get user's attributes from certificate;
    if the user attributes match the authorization rules
        if user is present on authorization list
            return GetData(key);
        else
            Deny access to the user;
        end if
    else
        Deny access to the user;
    end if

}
```

Code snippet A.6: GetData pseudocode

A.7.6 Applicability

Adopt a blockchain-based storage model when data is required to be stored immutably with distributed topology.

Traditional storage models are mostly built around a centralized architecture such as a relational database. Relational databases do not store data in an immutable manner and are not in line with all the policies and regulations pertaining to storing electronic health records.

The nature of blockchain technology is to store records immutably in an append-only format. Storing electronic health records in a blockchain network is mostly in line with the policies surrounding EHRs. The Health Professions Council of South Africa (HPCSA) stipulates that when health records are stored in an electronic format it should be done so in an append-only format (Health Professions Council of South Africa, 2016b). Copies of the records should be made and stored in different physical locations. The copies are used to detect tampering with electronic health records. Health records should also be kept for at least five years. Blockchain technology is aligned with these policies, as the data stored in a blockchain network is distributed across peer nodes situated in different physical locations. The POPI act, however, states that users should be able to request that their personally identifiable information (PII) is purged from a service (South Africa, 2013, section 24). As blockchain technology stores data immutably, it cannot satisfy this requirement.

Use a blockchain-based storage model to store a hash of the data and a reference to the location of the data.

Relational databases satisfy the personally identifiable information (PII) requirements by supporting the purging of records as stipulated in regulations such as the PoPi Act.

Blockchain technology can support the traditional centralized infrastructure by storing a hash of data that is contained in a relational database. This method of storing EHRs would enable stakeholders to run integrity checks on data stored in the traditional architecture. The hash of the data stored in the traditional storage model can be compared with the hash stored in the blockchain storage model. The two hashes should be identical to pass an integrity check. Blockchain technology can thus support the integrity of electronic health records stored in the traditional architecture. Refer to the blockchain-based audit log to keep track of data robustly.

A.8 Blockchain-based transaction

Transactions are used to add, update, or retrieve data from databases. Data transactions in this context also apply to the sharing of information between authorized parties.

A.8.1 Intent

The common transaction process used by enterprise systems relies on a centralized client-server model. Blockchain technology could provide a distributed and practically immutable audit log. Permissioned blockchain technology can replace the client-server model with a peer-to-peer model for transacting data.

A.8.2 Problem

The information flow of the blockchain-based transaction model is illustrated in Figure A.7. The data flow diagram is based on the authentication and authorization data flow diagrams. The transaction data flow diagram omits the authentication and authorization process to simplify the diagram. The process assumes that a user has been authenticated. Users from one organization can send a patient's EHR record to another organization, the EHR system encrypts the record and sends it to the blockchain client. The blockchain client forwards the encrypted EHR data to all of the organizational peers involved in the transaction. The respective peers store this transaction in their private state. This means that only the organizations involved in this transaction would have the EHR data stored in their peer's private state. The blockchain client then creates a hash of the transacted EHR data and broadcasts that to all the peer's public states. These peers also include peers from other organizations that are not a part of the transaction. This is to ensure a level of transparency and audibility across organizational bounds. The organizations' part of the transaction can then retrieve the EHR record from their

peer's private state. The sending organization could specify an expiration date for the transaction, meaning that the data would only be available to an organization for a specific period. After the period has expired, the data is automatically purged from the blockchain and only the hash of the transaction remains.

A.8.3 Pseudocode

The following section outlines a generalized transaction smart contract based on experiments conducted with Hyperledger Fabric. The TansactData method found in Code snippet A.7 outlines pseudocode to transact data between organizations that are part of the blockchain network. Retrieving transacted data could be achieved with the pseudocode function present in Code snippet A.8. The functions explained in this section could be ported to any permissioned blockchain network that supports smart contracts and certificate-based authentication.

```
Function TransactData(Transient data, Recipients recipients, Time
    TimeToLive) {

   Do sanitization check on the function inputs;
   Get the users certificate;
   Validate users certificate;
   Get user's identity from certificate;
   Get user's attributes from certificate;

   if the user attributes match the authorization rules
      if userID is present on authorization list
           for each recipient
             Set data expiration date;
             Encrypt data with recipient public key;
             Encrypt data with recipient peer public key;
             Send data to receipt's peer private state;
           next
           //Store on all peers public state.
           StorePublic(senderSignature, recipients, dataHash);
       else
           Deny further access;
       end if;

   else
       Deny access to the user;
   end if;

}
```

Code snippet A.7: TransactData pseudocode

```
response Function GetTransactedData(Key key) {

    Do sanitization check on the function inputs;
    Get the users certificate;
    Validate users certificate;
    Get user's identity from certificate;
    Get user's attributes from certificate;

    if the user attributes match the authorization rules
       if userID is present on authorization list
           Return GetPeerPrivateSate(key);
        else
           Deny further access;
        end if;

    else
        Deny access to the user;
    end if;

}
```

Code snippet A.8: GetTransactedData pseudocode

A.8.4 Applicability

Use the blockchain-based transaction model to transact private data in a peer-to-peer topology.

The traditional transaction model relies on a third party to handle private information. The information would be sent from the sending client to a centralized third part server back to the receiving client. This approach can increase the risk of a man in the middle attack.

Blockchain technology enables a sending client to send private information directly to the receiving client without relying on a third party to relay the information, thus enabling healthcare providers to transact a patient's health records without relying on a third party.

Use the blockchain-based transaction model as a peer-to-peer service to sync information across organizations.

Blockchain technology could be used as a sync service to transact information between organizational data stores. Coupling the blockchain-based transaction model with the authentication, authorization and audit log model would establish a robust sync service with full audibility across multiple organizations, which could include hospitals, private practices, pathology laboratories, medical insurance companies, pharmacies, auditors, medical boards, etc. This would, in turn, enable patients to visit any healthcare

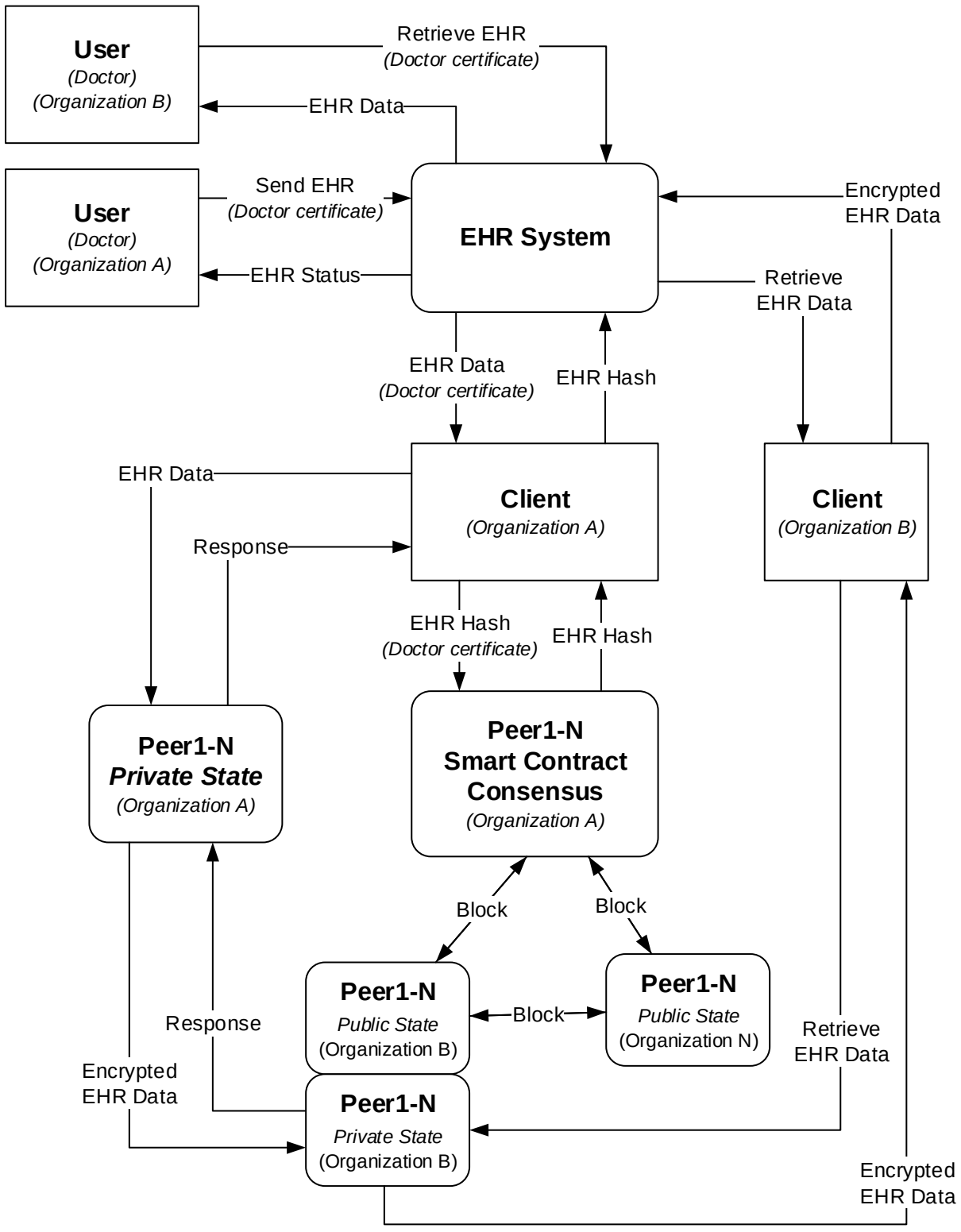

Figure A.7: Blockchain transaction data flow diagram

provider that is a part of the blockchain network. The patient could then provide authorization to the healthcare provider to sync his/her information with their data stores, essentially providing the healthcare provider with his/her electronic health record. The authentication, authorization, audit log model would be distributed across all of the organizations logically forming a single unified system. Blockchain technology could thus provide a robust semi-decentralized transaction model.

A.9 Conclusion

The report briefly outlined the current state of EHRs and how blockchain technology could enhance the transitional EHR infrastructure. Electronic health records are contentiously facing cyberattacks and it has become clear that the traditional infrastructure can no longer guarantee the safety of electronic health records. Blockchain technology has been evolving and can now be classified into three distinct types known as public, permissioned and private. The various types of blockchain technology were compared in tables. Experiments were conducted with various blockchain technologies to establish the advantages and disadvantages associated with the technology. Permissioned blockchain technology was identified as one that could improve the traditional EHR infrastructure. A permissioned blockchain technology known as Hyperledger Fabric was chosen to execute specific EHR experiments. The experiments revealed that permissioned blockchain technology could enhance various aspects of the traditional EHR infrastructure. These aspects include the authentication, authorization, audibility, storage and transaction models related to EHR systems. These aspects were generalized and outlined in various forms in this report. The blockchain-based enhancements were also discussed in terms of structure, pseudocode, and applicability. The findings were based on specific functions built into Hyperledger Fabric and these functions could be incorporated into any permissioned blockchain technology provided that technology supports smart contracts and certificate-based authentication and is open to extension. Hyperledger Fabric is fully open-sourced and can be tailored to fulfill the complex requirements of enterprise use-cases. The recommendation is to use the Hyperledger Fabric framework as a starting point. This report concludes that permissioned blockchain technology could enhance the traditional electronic health records infrastructure.

Appendix B

Publication resulting from the study

Adlam, R., Bertram, H. (2019). A Permissioned Blockchain Approach to the Authorization Process in Electronic Health Records. In *International multidisciplinary information technology and engineering conference*. IEEE.

Abstract

In recent years, electronic health records (EHRs) have been subject to data breaches. The result of these data breaches indicates that the current EHRs infrastructure is no longer suitable for safeguarding health records. Blockchain technology has been evolving and could improve the current EHRs infrastructure. Literature suggests that EHRs commonly utilize a role-based access control (RBAC) model. Permissioned blockchain technology could improve the authorization model by leveraging smart contracts and attribute-based access control. Hyperledger Fabric has been identified as a permissioned blockchain technology suited towards use-cases that require privacy. An experiment has been conducted with Hyperledger Fabric to illustrate the benefits permissioned blockchain tech- nology could provide to EHRs. The result of the experiment revealed that permissioned blockchain technology could improve the authorization model used by traditional EHRs.