Master of Science in Software Engineering
June 2019

Exploration of using Blockchain technology for forensically acceptable audit trails with acceptable performance impacts.

Abedallah Sobeh

Faculty of Computing, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden

This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfilment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 20 weeks of full time studies.

The authors declare that they are the sole authors of this thesis and that they have not used any sources other than those listed in the bibliography and identified as references. They further declare that they have not submitted this thesis at any other institution to obtain a degree.

Contact Information:
Author:
Abedallah Sobeh
E-mail: abso13@student.bth.se

University advisor:
Associate Professor Emiliano Casalicchio
Department of Computer Science

Faculty of Computing
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

Abstract

In this work, we will test the possibility to use Blockchain to preserve data such as logs. Data inside Blockchain is preserved to be used as digital evidence. The study will examine if Blockchain technology will satisfy the requirement for digital evidence in a Swedish court. The study will simulate different test scenarios. Each scenario will be tested on three different hardware configurations. The test has two main categories, stream test and batch test. In stream test, we test performance impact on different systems in case each log is sent in a separate block. While in batch test, we have two categories batch with data and batch without data. In this test, we simulate sending 80GB of data each day. In total we send 80GB of data, but the difference here is that we change the time between each block and adjust the size of the block. In our tests, we focused on three metrics: CPU load, network bandwidth usage and storage consumption for each scenario. After the tests, we collected the data and compared the results of each hardware configuration within the same scenario. It was concluded that Blockchain does not scale up in stream mode, and it is limited to ten blocks/s regardless of hardware configuration. On the other hand, Blockchain can manage 80GB of data each day without stressing system resources.

Keywords: Blockchain, performance, digital evidence

Sammanfattning

Det följande arbetet undersöker vilka möjligheter som Blockchain har som ett verktyg för att spara och bevara känslig data, för att kunna användas som digitala bevis. Dessutom ska studien undersöka giltigheten av Blockchain-tekniken som bevis i domstolen. Studien bygger på ett test som simulerar 15 scenarier med tre olika hårdvarukonfigurationer. Testet delas upp i två huvudkategorier, stream test och batch test. I stream testet, testar vi prestationseffekten på olika system när varje logg skickas i ett separat block. Under batch testet har vi två underkategorier vilka är batch med data och batch utan data. I batch testet simulerar vi att skicka 80 GB data varje dag. Under batch testet har vi dessutom testat att ändra på tiden mellan varje block generering och även justerat blockens storlek. I våra test har vi fokuserat på tre mätvärden: CPU-belastning, användning av nätverksbandbredd och konsumtion av lagringsutrymmet i varje scenario. När samtliga test slutförts, började vi med datainsamling och jämförde resultaten från varje system inom samma scenario. Slutsatsen är att Blockchain inte skalar upp i stream testet, då max antal block som skapas och skickas till data-noder är begränsat till tio block/sek, oavsett hårdvarukonfiguration. Däremot, vid batch testet , kan Blockchain hantera överföring av 80 GB data varje dag (24 timmar) utan att anstränga systemsresurser.
Nyckelord: Blockchain, prestanda, digitala bevis

Acknowledgments

This work would not have been possible without the help and support of my supervisors at BTH and City network. I am very grateful for the help and support from Associate Professor Emiliano Casalicchio from BTH and Kim Hindart from City Network. I am especially indebted to Jur. Dr Jonas Ekfeldt who helped in explaining Swedish laws and regulations related to our work.

Nobody has been more important to me in the pursuit of this project than my family. I am very grateful to my parents, for their love and guidance. I am thankful to my loving and supportive wife, Fatma, and my beautiful two children, Hanan and Reem, who provide constant inspiration

List of Tables

5.1 Scenario 1 test results 23
5.2 scenario 2 test results 24
5.3 scenario 3 test results 24
5.4 scenario 4 test results 24
5.5 scenario 5 test results 24
5.6 scenario 6 test results 25
5.7 scenario 7 test results 26
5.8 scenario 8 test results 26
5.9 scenario 9 test results 26
5.10 scenario 10 test results 27
5.11 scenario 11 test results 28
5.12 scenario 12 test results 28
5.13 scenario 13 test results 28
5.14 scenario 14 test results 29
5.15 scenario 15 test results 29

List of Figures

2.1 structure of Blockchain 4

4.1 Framework workflow 18

5.1 Overall CPU utilization in streaming mode 25
5.2 Overall CPU utilization in batch mode with data 27
5.3 Overall CPU utilization in batch mode without data 29

A.1 CPU utilization For SUT1 in scenario1 47
A.2 Network utilization For SUT1 in scenario1 48
A.3 Storage consumption For SUT1 in scenario1 48

Contents

Abstract i

Sammanfattning iii

Acknowledgments v

1 Introduction 1
- 1.1 Motivation . . . 1
  - 1.1.1 Problem statement . . . 2
  - 1.1.2 Aim and objectives . . . 2
  - 1.1.3 Thesis questions . . . 2

2 Theory 3
- 2.1 Blockchain . . . 3
- 2.2 Hashing . . . 4
- 2.3 Public Blockchain . . . 4
- 2.4 Private Blockchain . . . 5
- 2.5 Mining . . . 5
- 2.6 Merkle Tree . . . 5
- 2.7 Blockchain applications . . . 5
- 2.8 Digital evidence in the Swedish laws . . . 6

3 Related Work 9

4 Method 11
- 4.1 Planing Phase . . . 12
  - 4.1.1 Deviation from the plan . . . 12
- 4.2 Literature review . . . 13
- 4.3 Validity . . . 13
  - 4.3.1 Internal validity . . . 13
  - 4.3.2 External validity . . . 13
- 4.4 Experimental design . . . 14
- 4.5 Limitations . . . 14
- 4.6 Execution . . . 15
  - 4.6.1 Experimental environment . . . 15
  - 4.6.2 Proposed framework . . . 15
  - 4.6.3 Framework workflow . . . 17
  - 4.6.4 Testing scenarios . . . 18

4.6.5 System under test (SUT) . . . 19
4.6.6 Data collection . . . 20
4.6.7 Testing preparations . . . 21

5 Results **23**
5.1 Streaming mode . . . 23
5.1.1 Scenario 1 . . . 23
5.1.2 Scenario 2 . . . 23
5.1.3 Scenario 3 . . . 24
5.1.4 Scenario 4 . . . 24
5.1.5 Scenario 5 . . . 24
5.1.6 Overall results for streaming mode . . . 25
5.2 Batch mode with data . . . 25
5.2.1 Scenario 6 . . . 25
5.2.2 Scenario 7 . . . 25
5.2.3 Scenario 8 . . . 26
5.2.4 Scenario 9 . . . 26
5.2.5 Overall results for batch mode with data . . . 26
5.2.6 Scenario 10 . . . 27
5.3 Batch mode without data . . . 27
5.3.1 Scenario 11 . . . 27
5.3.2 Scenario 12 . . . 28
5.3.3 Scenario 13 . . . 28
5.3.4 Scenario 14 . . . 28
5.3.5 Scenario 15 . . . 29
5.3.6 Overall results for batch mode without data . . . 29

6 Analysis and Discussion **31**
6.1 Blockchain and ethic . . . 31
6.2 Streaming mode . . . 31
6.2.1 Scenario 1 . . . 31
6.2.2 Scenario 2 . . . 32
6.3 Scenario 3 . . . 32
6.3.1 Scenario 4 . . . 33
6.3.2 Scenario 5 . . . 33
6.3.3 Overall results for streaming mode . . . 34
6.4 Batch mode with data . . . 34
6.4.1 Scenario 6 . . . 34
6.4.2 Scenario 7 . . . 35
6.4.3 Scenario 8 . . . 35
6.4.4 Scenario 9 . . . 36
6.4.5 Scenario 10 . . . 36
6.4.6 Overall results for batch mode with data . . . 37
6.5 Batch mode without data . . . 37
6.5.1 Scenario 11 . . . 37
6.5.2 Scenario 12 . . . 37
6.5.3 Scenario 13 . . . 38

6.5.4 Scenario 14 . . . 38
6.5.5 Scenario 15 . . . 39
6.5.6 Overall results for batch mode with data . . . 39

7 Conclusions and Future Work **41**
7.1 Can Blockchain be used as evidence? . . . 41
7.2 Computer resource usage in stream mode . . . 41
7.3 Computer resource usage in batch mode . . . 41
7.4 Future work . . . 42
7.4.1 Network optimization . . . 42
7.4.2 Real-world test . . . 42

References **43**

A Supplemental Information **47**
A.0.1 Collected data representation for SUT1 in scenario 1 . . . 47

Chapter 1

Introduction

A tremendous amount of data traces is found on electronic devices such as computers and mobile phones. These data traces need to be interpreted in order to be used as digital evidence. Digital forensic is a process that is used to interpret and uncover these electronic data into evidence[29]. For a court to accept a digital forensic report as evidence, digital forensics process should follow rigorous procedures. A vital procedure is evidence acquisition and preservation as it is essential to be both deliberate and legal. Otherwise, the evidence will not have a value in courts. It is very critical to be able to document and authenticate the chain of evidence[14, 19]. One method that is proven to be robust and trusted in securing digital data, is the use of Blockchain. Blockchain-based solution helps in tracing the digital forensic chain of custody. This is achieved by creating a secure digital ledger that records and stores events and records. This means that it creates an unimpeachable audit trail within the Blockchain network[11].

1.1 Motivation

An essential process in digital forensic process is data collection. By data collection, we mean the process of securing information on a digital media that are intended to be used as evidence. To the Swedish Police, this process means to confiscate the digital media for investigating. The process is documented to ensure data integrity[15]. During the investigation, the investigated media is held as evidence, and the owner will not be able to use or claim it until the end of the investigation. This procedure causes losses for web hosting companies as multiple servers can be confiscated on an ongoing investigation.
A very popular incident was the *PRQ* case in 2006. In this case the police forces were investigation an illegal torrent sharing website, *"The Pirate Bay"*. To investigate the site, multiple servers were confiscated by the police. Some of these servers hosted other services than the investigated site. This resulted in losses for the hosting company and other companies who are using *PRQ* services. *PRQ* lost its clients as well as the trust in the company. The clients of *PRQ* suffered as well from losses when their services went offline. *PRQ* claimed 600,000kr as compensation, but the court rejected their claims[4, 12, 5]. We think that theses Losses could be avoided if a robust logging solution were implemented. This will give the Police the ability to investigate and build a case without having to confiscate the servers of *PRQ*.

1.1.1 Problem statement

Digital investigation is not limited to crime and Police investigations. Sometimes corporations need to investigate incidents like data breach or a hacking incident. The purpose of such investigation is to find out what happened, where and when did it happen, why did it happen and who did it. In such cases, it is not necessary to involve police forces. If the involved parties agreed to a logging solution, this would help in determining the cause of the problem without the need for a police investigation or the need for confiscation hardware and servers.
The problem here is that implementing such a system comes with a price. The price for such a system is CPU usage, network bandwidth usage and storage consumption for storing block. The parties should be aware of these costs before the implementations. This study will point out these costs so that each party will be aware of it before the implementation of Blockchain for log preservation.

1.1.2 Aim and objectives

Implementing Blockchain in digital forensics systems comes with performance impacts on the system. We found some articles and studies about using Blockchain in digital forensics, but we did not find reviews about the performance impacts on Blockchain systems. However, there are some tools (such as IBFT) to benchmark Blockchain networks, but these tools do not address the performance impact of implementing Blockchain[24]. This study will address the performance requirement as well as the storage requirement of Blockchain in terms of storage capacity that is needed in the different implementing scenario. We define the goal of this work as the following:

- studying the possibility to use Blockchain from the perspective of law.
- identifying a framework for implementing Blockchain.
- studying the performance impact of using Blockchain in the following cases:
  - storing and sending log information (data) in the Blockchain alongside the hash values or:
  - sending log information to a separate storage solution and sending only hash values in Blockchain.
- compare the performance against storing the hash on every log update or making archives on a schedule and storing the hash of the archive.

1.1.3 Thesis questions

This thesis will address the following research questions:
RQ1) Is Blockchain a valid method for preserving digital evidence and will the courts accept the Blockchain as a valid way of ensuring data integrity from the perspective of digital evidence?
RQ2) What are storage and performance resources that Blockchain solution requires for storing digital evidence?

Chapter 2

Theory

2.1 Blockchain

The Blockchain concept was introduced to the world after the invention of Bitcoin in 2008. It is a growing technology that will affect every industry, such as law, media and arts. By 2025 it is expected that Blockchain will become a mature technology with a huge number of users. Since 2017 a remarkable increase of interest in Blockchain has been noticed and searching for Blockchain in Google has been rapidly growing. This resulted in attracting more researchers and communities to study and develop Blockchain technology. There are multiple benefits of Blockchain such as trust, cost-saving and efficiency but there are some challenges that researchers are focusing on such as scalability[8].

Blockchain can be described as a data structure of a series of connected nodes. These nodes are called blocks, and every block has information to track changes that happen on a per-to-per system, which mean that all participants can communicate directly without the need to a third party. Timestamps and hash values inside each block ensure the integrity of data. Each block is linked and depends on the previous block to form a chain. This results in a chain of a permanent and irreversible history of blocks that can be verified by viewing the data and calculating hash values. This means that Blockchain can be used to ensure the integrity of the data. This is achieved in a per-to-per system as both ends can calculate hash values of a given set of data. If both nodes calculate the same value, the new block will be added to the chain. This means that both nodes have agreed and confirmed the validity of the new block. This process ensures the integrity of each block of the data[16, 9].

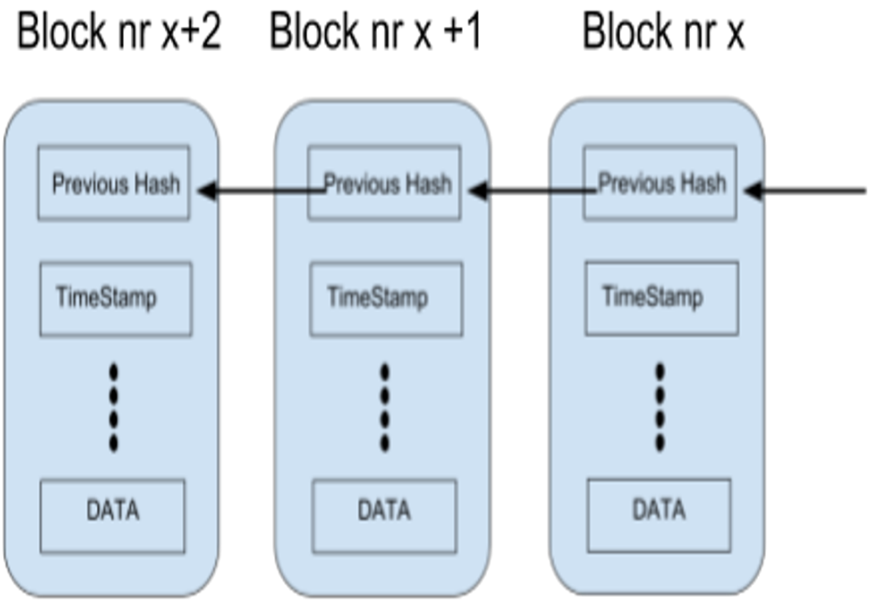

Fig. 2.1: *structure of Blockchain*

2.2 Hashing

Hashing is the process where the hashing function maps original data to a fixed-size string. To consider a hash function as a good and secure hash algorithm it should meet some requirements[27, 7]:

- The output of the function should have a fixed length(sha256 has the length of 256).
- Any change in the input data will result in totally different output.
- Using the same data as input will generate the same output (hash).
- The function should be irreversible, this means that we can not find or generate input data by knowing the hash (output of the hash function cant lead to the input of the function).
- The hashing function should be fast and should not intensively use CPU.

2.3 Public Blockchain

In public Blockchain it is not required to get a permission to be a part of the chain. as the mane suggests This type of Blockchain is not owned by anyone. The chain is shared publicly and each node can have a copy of the chain. This allows any part of the chain to Contribute to the chain without any permission. Each node in the network can perform a validation of the chain[8].

2.4 Private Blockchain

As the name suggest, this type of Blockchain is private and is closed to a group of members. The groups may contribute to the chain, but the chain remains inaccessible for the outside. This kind of Blockchain is useful for organizations who need to share a secure ledger inside the organization itself[8].

2.5 Mining

Each peer in a Blockchain network is called a node, and any node in a public Blockchain network can create new blocks. The process of creating a new block is called mining, and the creator of the block is called a miner. Mining a block can be performed using minimal hardware, but the difficulty of creating new blocks increase with time[22]. This means that it is harder and harder to mine new blocks.

2.6 Merkle Tree

Merkle tree is a binary tree of hashes. Each node in the tree represents a hash value of the data block. In the tree. Each parent in the tree represents the hash values of the child nodes. Hashing is done in a bottom-up approach and in an append-only manner[21]. This approach is used to validate the tree based on the history of the performed operations. As the history of the tree can be validated by the following and recalculation hashes. The tree can be used to validate the data within the tree.

2.7 Blockchain applications

As mentioned before, Blockchain is a fast-growing research field. There have been a considerable number of researches about using Blockchain for different applications. According to Z. Zheng et al Blockchain application can be categorized into five categories: finance, IoT, public and social services, reputation system and security and privacy[28].

- Finance: A huge effect is brought to the traditional industry after introducing Bitcoin and hyperledger. The benefits of such a system lies in eliminating the need for having a third-party such as banks in the trading process. This reduces the time for the processing and saves fees that go to third-party. All this can help to improve and revitalize the traditional industry.

- IoT: The sector of IoT is a growing sector, and it has many applications. Blockchain can help to improve this sector. A huge concern in IoT community is safety and privacy. By integrating Blockchain with IoT devices, it will help in proving the manufacturing provenance without the need for a third-party authority to confirm it.

- Public and social services: Blockchain has a wide range of applications that can be used in public and social service. It can be used in land registration,

energy saving, education, etc. A possible application is to use it for registering marriages. Another application is to use it for patent management and the registration of the taxation system

- Reputation system: Reputation is a measure of how much a community trusts you. Especially in e-commerce is it essential to have gain customers trust and to have a good reputation. The majority of e-commerce sites allow for evaluation and feedback after an online purchase. This would help them to develop their products, and good reputation encourages other costumers on buying from the same service provider. Some companies may enrol a huge number of fake customers feedback to gain a better reputation. If Blockchain is integrated into customer evaluation system, it will help to solve this problem. This can be achieved as Blockchain will ensure the immunity and the integrity of the feedback. In other words, this will prevent false and fake feedbacks.

- Security and privacy: There is an increasing number of electronic devices. These devices are vulnerable agents malware attacks. There is a number of anti-malware software and filters to help in preventing these attacks. The problem hers is that each vendor has its own database for detecting such attacks. This databases are vulnerable against malware attacks and implementing a decentralized Blockchain system will prevent affecting theses databases by malware attacks.

2.8 Digital evidence in the Swedish laws

The Swedish code of judicial procedure(rättegångsbalken), is the source of Swedish evidence law. The code entered into force by 1942, and its regulation has many aspects based on preparative legislative work from 1928. This code regulates most aspects of both civil and criminal cases. In regards to evidence, the code states five different types of evidence[6]:

- Witnesses
- Examination of Parties and of Aggrieved Persons not being a party
- Documentary evidence
- Views
- Experts.

The code does not come across or regulate digital evidence. The Swedish code does not provide the judges with proper tools to assess digital evidence. Here the evidence presentation is based on the principle of free evidence[3]. This principle state that:

- the parties are free and without any restriction can present any evidence regardless of the shape or the form of the evidence.
- The court has the freedom the evaluate, accept or refuse the evidence without the need to abide by any legal rules.

This means that anything that can help in a case can be presented in a court regardless of the form, type or shape. Only the court can decide if the evidence can be used or not in the case.

Chapter 3

Related Work

Similar work has been done in previous years. Mattias Scherer in his paper "Performance and Scalability of Blockchain Networks and Smart Contracts" compares public Blockchain with permissioned Blockchain[25]. Scherer even examines the possibility to use permissioned Blockchain to replace the old systems in financial institutes using stress tests. Scherer conclusion is that *Hyperledger Fabric* is applicable and much faster than other existing solutions such as *Ehereum*. In the paper, Scherer has never discussed the legal aspects of using such a system. Another thing that Scherer missed is to compare the performance where the data is a part of the Blockchain compared to sending the data to a separate storage solution.

In their paper Park Y., Park H. and Huh E. propose as Blockchain based framework[20]. In the article, they investigate the usage of permission Blockchain based integrity management system to log activities in cloud environment for forensics use. The proposed system guarantees data integrity and works faster than other existing systems. The missing part here is a full assessment of the system in terms of CPU usage and storage usage, as well as they send data to a separate database log system.

Dhumwad et al. proposed a new protocol for money transfer in a peer to peer system[21]. The suggested protocol is based on Merkle tree and SHA256 hashing technique. Although the work discusses the protocol, the impact and resource usage of such a protocol are not discussed in the article. Here they use Java as a programming language. In our work, we use the Merkle tree and SHA256 in our proposed framework. The difference here is that we use Python as a programing language while they use Java. The use of different programming language may affect resources consumption. In our work, we point out the requirement and resource consumption that is used by our framework.

Although we did find articles about the overhead and the consumption of the resources when implementing Blockchain applications, there were many articles about the applications of Blockchain. In their paper M. Nofer, P. Gomber, O. Hinz and D. Schiereck introduced multiple applications of Blockchain in the field of Cryptocurrencies, Securities issuance and Insurance[17].

Chapter 4

Method

Research is the process of identifying a problem, formulating a research question and then engaging with the subject by gathering evidence from the data to support and answer the presented knowledge. In the previous sections, we defined the problem and research questions[13]. In this section, we will present and discuss the methods that we will use to answer our research questions.

To assess the validity of Blockchain to be used in a court, we will study the legal requirement for acceptable digital evidence. We will conduct a literature study of the legal requirement of digital evidence. It is also planed to interview digital forensic analysts working in Swedish authorities such as Police force and National Operational Department(NOA). Here we will be asking questions and discussing within the targeted group. Interview is a very popular method in almost every professional area as it is cost effective and it stimulates open discussions[10, 23]. The advantages of using this method are that it is less expensive and needs less time to be completed (compared with other methods). The only disadvantage of this method is that the results may not be accurate. The cause of that is that the results may reflect the opinions of those who have the majority inside the group[23]. We think that this disadvantage is irrelevant in our case as the targeted group consists of professionals who are experienced and worked in the field of digital forensics. The answers and the reflection from the group are based on their own experiences, previous work cases and the law. Here we will limit us to the Swedish law.

To assess if Blockchain is practical and usable, experiment method with comparative study design will be used. This method is the most appropriate method for studying the impact and/or the effectiveness of using different models[23]. Each model is tested on a set/group of data, and the results from each model will be compared to a well-defined baseline. The advantage of using this model that gives us the ability to measure the changes and to assess the impacts of the implementation[23]. Action research method would fit in our case as this method is used when we want to improve our understanding or develop our learning about a case. This method was not used as it does not fit if the purpose of the research is to draw comparison, show statistical correlations or effect relationship[13]. To answer the second research question, we will study :

- the storage requirement and the performance impacts of sending digital evidence data, (log streams or log batches in our case) as a part of Blockchain against sending only the hash value in Blockchain while the data will be sent

to a separate log server.

- the performance impact of sending log streams against sending a batch of log entries through the Blockchain

In our study we define the baseline to compare performance impact and storage requirement as a logging solution where the logs are saved locally on the server. This research will study the overhead introduced by implementing Blockchain. Different study cases will be defined later.
The study will be based on common and widely used open source components to ensure repeatability and testing should be run and deployed in a public cloud environment.
For testing, we will use a set of log files that are provided by the company (City network). We will start by discussing (together with the company), what tools and what cloud environments to use as a base to our work and tests.
The next step is to deploy Blockchain in two different public cloud tenants (which we chose in the previous step), to simulate a shared Blockchain between 2 parties.
The last step is to document and compare storage requirement and the performance impact of implementing the different methods mentioned in the previous section,(Aim, objectives and thesis question(s)).

4.1 Planing Phase

The first month of the project was allocated for a literature study on Blockchain technology and digital evidence regulations in Sweden. This month was used to find suitable tools and software for the PoC (proof of concept). Resources that we used in our research are books, articles and websites and blogs for people and researcher with experiences and/or are working in the field Blockchain. We planned to interview digital forensics employees from the Swedish Police, Financial Inspection authority and National Operational Department, (NOA). In the middle of the first month, we started contacting these authorities to Nominate some employees to the interview. These authorities were contacted by telephone and by e-mail. The purpose of e-mail contact was to give them a detailed overview of the research and to inform them about the nature of our questions. The second month was spent on preparing for the interview. In the same time, we worked on Framework design for the PoC. The third and fourth-month were devoted to conducting tests on the framework documenting the results and analyzing the results. The remaining time was used for writing the thesis and controlling the collected data.

4.1.1 Deviation from the plan

We planed for interviewing digital forensic employees from Swedish authorities such as the Swedish Police. We could not go through with this plan as the authorities had limited resources and they could not allocate some of their employees for the interview. The alternative that we chose, was to contact Jur. Dr Jonas Ekfeldt. Ekfeldt is specialized in training lawyers and police employees in the field of digital evidence. Unfortunately, Dr Ekfeldt did not have time for an interview. Through

e-mail communication, Dr Ekfeldt helped us by clarifying regulations for evidence is Sweden.

4.2 Literature review

To answer this thesis questions, it was essential to identify the related work of other researchers in the field of Blockchain. This would help us to understand other researchers approach, and it helped us to understand how they addressed the issue. Through scientific databases such as (IEEE explore and Google Scholar), we were able to identify some related work. Although other researchers addressed Blockchain from different perspectives, it was beneficial to see other approaches. Using various and specified search queries (such as Blockchain AND (performance OR CPU consumption)), we were able to identify a massive number of articles. The results were narrowed by evaluating based on the title and the abstract of the article. After the evaluation of the articles, we had a smaller set of articles that we used in our theory and related work.
Regarding the legal part of this work, we chose to use the english translation of the Swedish regulation of evidence as the source to cover the legal part of this work. This translation is published on the government website.

4.3 Validity

In this section, we will discuss all the measures that we took to ensure the validity and the repeatability of this work. Regarding the repeatability, we had clear rules in choosing tools and test environment. Every tool and library in this project are open-sourced and can be accessed by anyone. This ensures that everyone can use the same result using the same tools. On the other hand, we had a well-defined hardware specification (in terms of CPU, Ram and OS). A set of measures were taken to ensure that the results reflect only resource usage that is utilised by the framework. These Mesures are described under the section *testing preparations* 4.6.7.

4.3.1 Internal validity

Together with City Network's experience in hosting and cloud infrastructure, the environment and the chose of tools was made. The collaboration with Miljödata could Is considered to be an advantage and strengthen the experiment. The foundation of this work is the use of open source tools. All of City network's infrastructure and cloud environment are built on open-source tools. City Network uses OpenStack for its infrastructure, which is open-source software for creating private and public clouds.

4.3.2 External validity

Although the experiment is conducted on a cloud infrastructure that is owned by City Network, the results can be generalised. This work targets company and institutions that are interested in using Blockchain technology for preserving data. If

the requirements of Blockchain users are within the limit and the scope of this work, they will get similar results. On the other hand, if 80GB/day or 10 logs/second does not satisfy the need of the user, We think that further studies (based on the requirements of the user) should be done.

4.4 Experimental design

All experiments and tests on this thesis will be conducted using open-source tools and software. The goal of open-sourcing is to ensure the repeatability of the results of the research. The main idea is to implement existing Blockchain software to conduct our tests. After a month of researching, we found that the existing software lacks essential features. These features are the size of the block as it is limited to a specific size[1]. The second feature is the ability to search for a particular block as Blockchain does not support queries and will not be replacing the traditional database[26]. Search functionality is critical, and recently Google is investing in developing a new search tool for Blockchain, *"ETL"*. This tool will allow the uses to search more effectively for data and transactions[2].

To overcome the limitation in block size and for better search functionality, we decided to implement our version of Blockchain. This implementation will also be based on open-source tools and software. Later on, details about the application will be described.

4.5 Limitations

As the legal requirements for acceptable digital evidence differ from county to another, we will limit our study to Swedish laws and regulations. Here we will analyze regulations for evidence used in Swedish courts, as well as we will contact a lawyer with expertise in the field of digital investigation.

The proposed framework will be limited to five functions, monitoring, creating, notifying, sending and validation. In case of streaming mode, the system will monitor all the changes on a specific file, and in batch mode, it will wait for a system signal upon creating a new file. After that, a new block will be created, all nodes will be notified, the block will be sent, and each node will validate the block. Data extraction from the block will not be covered in this work as it does not add overhead to the process. The reason for that is the extraction of data does not happen very often as generating new logs. But we tested data extraction and the data in the blocks were identical with the original data.

4.6 Execution

4.6.1 Experimental environment

In this section, we will briefly describe the tools, commands and technology that were used to assist in this research.

CityCloud: is a cloud service provided by City Network. It is based on OpenStack, and the servers are located all over the world. Citycloud provides infrastructure for its users to be used for deploying servers and services.

Ubuntu server 18.4 LTS: is a Unix-like operating system. The operating system is based on Debian, and it is used by many corporations for its simplicity and its features.

Python3: is a powerful programming language that is open-sourced with a generous number of libraries and a very supportive community.

MariaDB is a database server that is based on Mysql and many service providers around the world have used it. MariaDB is an open-source relational database server that provides SQL interface for accessing data.

Sar: is a Linux tool that is used to collect, report and save system activities.

Pidstat: is a Linux tool that reports statistics for a given task that are running on a Linux system.

Du: is a Linux command that is used to estimate the size of a given file on Linux system.

Dd: is a Linux command that is used to copy and convert files. It takes an input and an output file as parameters, and based on other parameters (such as the block size) it copies from the input file to the output file.

4.6.2 Proposed framework

For the purpose of designing the framework, we used the following tools:

- Programming language: Python 3
- Non-standard libraries: Flask
- Data base: Mariadb

The implementations of existing Blockchain tools and applications do not support all the features we need for our testing. These features are the ability to search for a specific block and the limitation and restrictions on block size. For these reasons, we

decided to implement a new framework that supports our needs. In our framework Python 3 is used as a programming language. Flask is used as a micro web framework. Flask is used as it is modular and it gives us a base for future development. As for now, it is used to manage connections between master-node and client-node. Https protocol is used for the communication between master-node and client-node because it is more secure. This ensures that data sent between the nodes cannot be sniffed or changed in the sending process.
Authentication process and chain management is processed through flask by calling different python methods and routines. Accessing Blockchain means to search, verify and to download data from blocks. This does not give the ability to change the content of the chain.
The proposed framework is based on the Merkle tree, where each block(node) is dependent on the previous block. Whenever a new data entry is signalled, the framework will generate a new block based on the has of the data and the hash of the previous block. This will result in a new block with the necessary information to validate itself and to trace changes in the block(if it has been manipulated). After the creation and the validation of the block, a new entry will be created in the associated database. The database is used to save information about the block such as timestamp, the group, the creator and the status(valid or not). The database is used for searching a specific block using the properties and the characteristics of the block. An example could be to search for events that have happened under a given date. Although database can be changed or manibulated if a user had access to the database, but all results from database queries are verified against the original Merkle tree. In case database is manipulated it can be easily regenerated from the original Merkle tree.

Our framework can be configured through a configuration file. Both master and client have a configuration file. In the master-node we have the following options to configure:

```
[group]
    name = <group name>
    [[members]]
        member1 = <Member name or id>

[member_info]
    [[Member name or id]]
        location = <Location>
        server_ip = <Ip address>
        server_port = <PORT>

[chain_config]
    type = <file | stream >
    owner = <block generator name or id>
    group = <group name or id>
    file_path = <Path to log file>
    dump_path = <Path for storing blocks>
```

```
    pid_file = <Path for saving main program pid>
```

Listing 4.1: *master-node configuration*

In our configuration in listing 4.1 we configured Master-node to communicate with one client, but it is possible to configure multiple nodes. In client-node, the configuration file is much simpler. It contains the necessary information needed for communication with master-node. The file has the following options to configure:

```
[config]
    server_ip = <master-node ip>
    server_port = <master-node port>
    local_port = <local port for flask>
    user_name = <user name for authentication>
    psk = <Password for authentication>
    storage_folder = <Path for storing blocks>
```

Listing 4.2: *client-node configuration*

Both Master and client have another configuration file. This configuration file contains necessary data to connect with the database server.

4.6.3 Framework workflow

In our implementation, the framework works in two different modes:

- stream mode
- batch mode

Both modes can be activated and configured through the configuration file. In the configuration file, the path for the targeted data is configured. Based on the configured mode the framework will work differently. In stream mode, the framework will monitor a file (tail) and as new log entries are appended to the file. The path for the monitored file is configured in the configuration file. When the program detects new log entries, it generates a new block. When a new block is generated, it will be added as the head of the chain. In batch mode the signal SIGUSR2 is used to notify when the file is ready. As the framework receives this signal, it will read the file name from the configuration file and create a new block for the file. The new block will be added to the head of the chain. In batch mode data file can be excluded and sent to a logging solution, but in stream mode the data is always a part of the block. This decision is based on the fact that the structure of the block is at least four times larger than a single log entry. Another reason is that the logs are a string of data and if every entry is saved to file, we will lose data integrity. The reason for that is that the hash of the file will not be the same as the one that is sent to the client in the block.

When a new block is added to the chain, database entry for the block is created. After that the master node will notify client/clients from the configuration file. This is done by sending a POST request to the client. The request form contains basic data about the block. This information is used by the client to create an upload

challenge. As the client receives this notification, it will send an acknowledgement to the master-node. The acknowledgement contains authentication data and a challenge string. When the master-node receives the challenge, it checks in the database for the user credentials. If the credentials match, it will upload the block to the node using the challenge that is sent by the client. The last step for the client is to check if the new block matches with the previously agreed challenges and it will check the validity of the hashes and the data. The node will recalculate the hash of the block by calculating the hash of the data and combining it with the hash of the previous block hash. If the hashes and the challenge match the new block will be saved, a new entry will be created in the database, and the master will be notified with acknowledgement of the block. Otherwise, the block will be ignored, and an invalid message will be sent to the master.

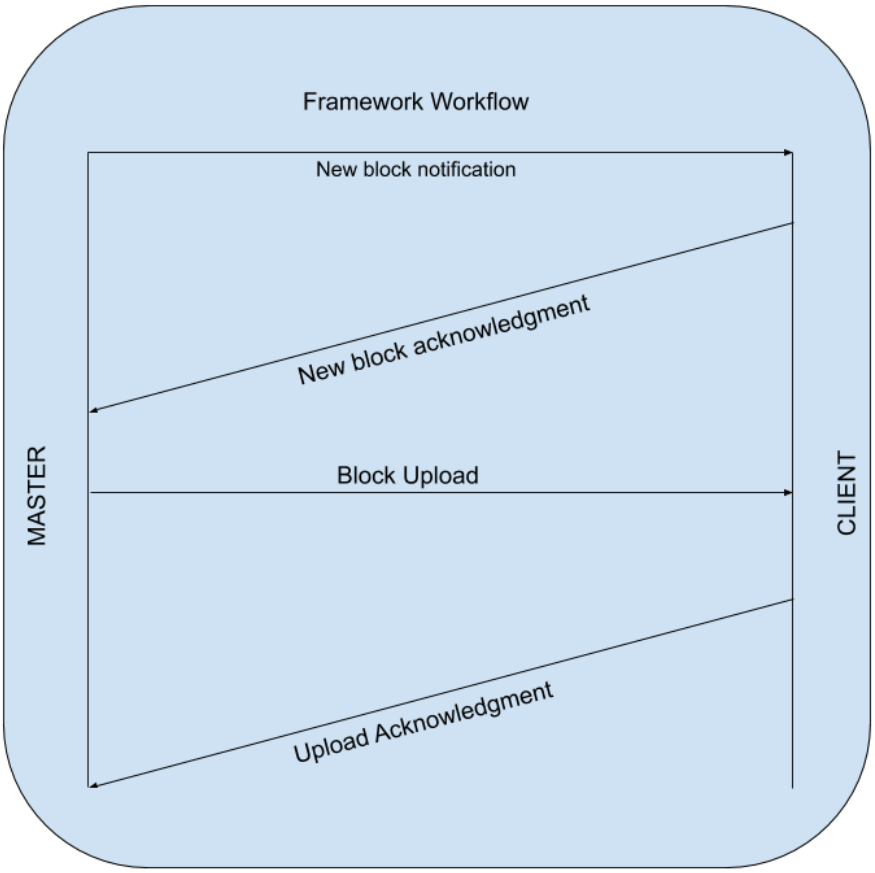

Fig. 4.1: *Framework workflow*

4.6.4 Testing scenarios

To test our framework, we chose to cover 15 different scenarios. After that, every scenario was tested on three different machines with different hardware. This resulted in performing 45 tests in total. Each case is tested for one hour. To ensure the validity of these tests, we repeated each tests ten times, and the average values were calculated. Based on the results we could estimate the average CPU usage, storage usage and used bandwidth for each case. The reason for choosing different cases is

to demonstrate the cost of implementing our framework based on the needs of its users. Test cases were divided into two main categories, *stream mode* and *batch mode* testing. In log streams testing we have the following test cases:

- scenario 1: sending 1000 logs each second to find the maximum number of blocks that can be sent in a second.
- scenario 2: sending 5 logs each second.
- scenario 3: sending 2 logs each second.
- scenario 4: sending 1 log each 0.75 second.
- scenario 5: sending 1 log each second.

According to City Network, as a small log corporation, they store about 40 GB of logs each day.These logs are saved in small zip files. We made our goal to test our framework by simulation 80 GB of data each day.
In log batch testing we have the following test cases:

- scenario 6: sending block with data of size 1MB each second
- scenario 7: sending block with data of size 10MB each second
- scenario 8: sending block with data of size 30MB each second
- scenario 9: sending block with data of size 60MB each second
- scenario 10: sending block with data of size 90MB each second
- scenario 11: sending block without data. Data size 1MB each second
- scenario 12: sending block without data. Data size 10MB each second
- scenario 13: sending block without data. Data size 30MB each second
- scenario 14: sending block without data. Data size 60MB each second
- scenario 15: sending block without data. Data size 90MB each second

4.6.5 System under test (SUT)

All test are conducted using three different system configurations, SUT1, SUT2 and SUT3. These systems have the following configurations:

- **SUT1** :
  - CPU configuration: Intel Xeon E3-12xx v2 @ 2 cores, 2.5Ghz
  - RAM: 4GB of ram
  - HDD: 100 GB
  - OS: Ubuntu server 18.04 LTS

- **SUT2** :
  - CPU configuration: Intel Xeon E3-12xx v2 @ 4 cores, 2.5Ghz
  - RAM: 8GB of ram
  - HDD: 100 GB
  - OS: Ubuntu server 18.04 LTS
- **SUT3** :
  - CPU configuration: Intel Xeon E3-12xx v2 @ 8 cores, 2.5Ghz
  - RAM: 16GB of ram
  - HDD: 100 GB
  - OS: Ubuntu server 18.04 LTS

All three systems are identical except for CPU and ram configuration. The goal here is to study the overhead introduced by Blockchain of different system configuration. Later on, in this work, we will refer to these systems as SUT1, SUT2 and SUT3.

Baseline configuration

As the study aims to identify the overhead introduced when Blockchain is used, we define each system as the baseline for its test cases. This means that SUT1 is the baseline for the tests on SUT1, SUT2 is the baseline for the tests on SUT2 and SUT3 is the baseline for the tests on SUT3. In other words, the outcome in the results chapter reflects resource usage on each system compared to the same system when Blockchain is not used.

4.6.6 Data collection

Usually, data collection should be randomized. In our case, the randomization does not affect the process. We are interested in studying the amount of data which we can manage in each test scenario. To conduct our tests, we used a bash script to generate logs. In case of stream mode we used a sample log from City Network to generate new log records. The script can be seen in 4.3

```
logfile=~/tempwork/forensic-blocks/logfile.log
: > $logfile
while true
do
      A=$(expr $A + 1)
        echo "type=AVC msg=audit(1553083167.916:13317719): avc:
      denied  { read } for  pid=2035 comm=\"handler237\" scontext=
      system_u:system_r:openvswitch_t:s0 tcontext=system_u:system_r:
      openvswitch_t:s0 tclass=netlink_generic_socket permissive=1" +
      $A >> $logfile
          sleep T
```

```
8       done
```

Listing 4.3: *master-node configuration*

The script starts by initializing file name. After that, it makes sure that the file does not have log entries and deletes all logs if it exists. In the next step, the script writes to log file the same log entry ending with a counter. The counter is used to trace the number of logs that are generated and added to the chain. In the last step the script sleeps for T seconds. The script runs on an infinite loop until it is exited manually.

In case of batch mode we used dd to generate different files with different sizes. the size depend on the test case it self. The script to generate these files can be seen in 4.4

```
1 dd if=/dev/zero of=logfile<nr>mb bs=1048576 count=<nr>
```

Listing 4.4: *master-node configuration*

The script generates a file with a size of *nr* MB. By changing the value of *nr* we generate a different file with different size. To generate a file with a size of 10 MB, we change set nr to 10.

4.6.7 Testing preparations

Both master and client were created in the same region. This means that if the master is created on Cityclouds servers in Karlskrona, the node will be created on Karlskronas servers. We chose to do that to eliminate any network latency or distortion that might happen in the test period. We also chose to set up database servers locally on each node. This will reduce the latency of SQL query.
Before performing any test, we restore both master and client to its original state. By that we mean to restore it to the state where database is emptied, and all blocks from previous tests are removed. With other words, we ran each test on a fresh installation of the system.

Chapter 5

Results

In this chapter, we will present the results of the testing scenarios we defined in the previous chapter. The results will be presented in the form of a data table. Each table will contain the average CPU load in percentage, the average network bandwidth used in kB/s and average storage consumption each second.

5.1 Streaming mode

5.1.1 Scenario 1

In this scenario, we tested all three systems to find the maximum number of blocks(logs) that we can generate and send to node computers each second. The results for test scenario 1 can be seen in 5.1.

System	Avg CPU usage %	Avg bandwidth kB/s	Avg storage consumption bytes/s
SUT1	39.7 %	99.9 kB/s	10100 bytes/s
SUT2	33.9 %	96.9 kB/s	10150 bytes/s
SUT3	17.5 %	97 kB/s	10140 bytes/s

Table 5.1: *Scenario 1 test results*

After conducting the test, we found that all the systems had the same limit. After conducting the tests we found 36000 saved block on the client side. This means that every system could generate and send a total of ten blocks/s.
A sample data representing can be seen in A.1, A.2, A.3. To reduce the amount of images we will only append the figures for SUT1 in scenario 1. The purpose of the images is to show the nature of the collected data.

5.1.2 Scenario 2

In this scenario, we tested all three systems to generate and send 5 blocks(logs) to node computers each second. The results for test scenario 2 can be seen in 5.2.

System	Avg CPU usage %	Avg bandwidth kB/s	Avg storage consumption bytes/s
SUT1	13.7 %	48.2 kB/s	5080 bytes/s
SUT2	24.4 %	49.5 kB/s	5085 bytes/s
SUT3	9.1 %	49.4 kB/s	5080 bytes/s

Table 5.2: *scenario 2 test results*

5.1.3 Scenario 3

In this scenario, we tested all three systems to generate and send 2 blocks(logs) to node computers each second. The results for test scenario 3 can be seen in 5.3.

System	Avg CPU usage %	Avg bandwidth kB/s	Avg storage consumption bytes/s
SUT1	8.89 %	19.59 kB/s	2041.36 bytes/s
SUT2	10.77 %	19.6 kB/s	2041.36 bytes/s
SUT3	3.7 %	19.53 kB/s	2041.36 bytes/s

Table 5.3: *scenario 3 test results*

5.1.4 Scenario 4

In this scenario, we tested all three systems to generate and send one block(log) to node computers each 0.75 second. The results for test scenario 4 can be seen in 5.4.

System	Avg CPU usage %	Avg bandwidth kB/s	Avg storage consumption bytes/s
SUT1	5.95 %	13.22 kB/s	1360 bytes/s
SUT2	7.24 %	13.4 kB/s	1360 bytes/s
SUT3	2.4 %	13.14 kB/s	1360 bytes/s

Table 5.4: *scenario 4 test results*

5.1.5 Scenario 5

In this scenario, we tested all three systems to generate and send 1 block(log) to node computers each second. The results for test scenario 5 can be seen in 5.5.

System	Avg CPU usage %	Avg bandwidth kB/s	Avg storage consumption bytes/s
SUT1	4.1 %	10 kB/s	1340bytes/s
SUT2	4.33 %	10.1 kB/s	1340 bytes/s
SUT3	1.88 %	9.9 kB/s	1340 bytes/s

Table 5.5: *scenario 5 test results*

5.1.6 Overall results for streaming mode

In each test scenario, each system had almost identical results in term of storage consumption and network bandwidth usage. The only difference was CPU utilization for each system. The following graph 5.1 summaries the results for CPU utilization in streaming mode.

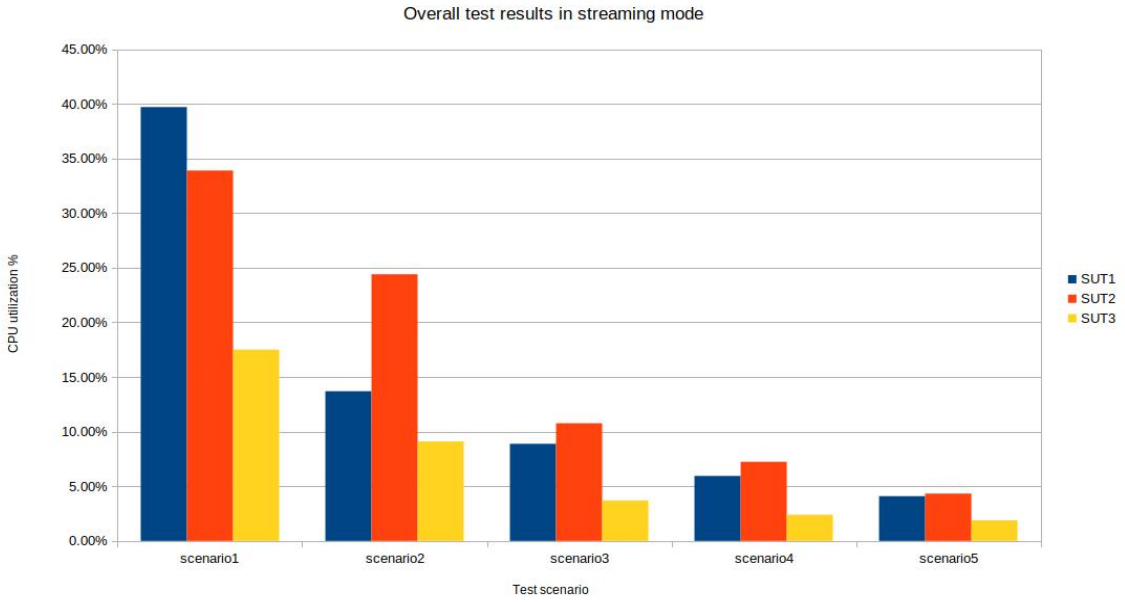

Fig. 5.1: *Overall CPU utilization in streaming mode*

5.2 Batch mode with data

5.2.1 Scenario 6

In this scenario, we tested all three systems to generate one block each second. The block will contain data with the size of 1MB. Each generated block will be sent to client-nodes. This means that we send a block with data size of 1MB each second. The results for test scenario 6 can be seen in 5.6.

System	Avg CPU usage %	Avg bandwidth kB/s	Avg storage consumption bytes/s
SUT1	6.1 %	1,039 kB/s	1,063,390 bytes/s
SUT2	8.1 %	1,040 kB/s	1,063,490 bytes/s
SUT3	2.9 %	1,038kB/s	1,062,889 bytes/s

Table 5.6: *scenario 6 test results*

5.2.2 Scenario 7

In this scenario, we tested all three systems to generate one block every ten second. The block will contain data with the size of 10MB. Each generated block will be sent

to client-nodes. This means that we send a block with data size of 10MB every ten seconds. The results for test scenario 7 can be seen in 5.7.

System	Avg CPU usage %	Avg bandwidth kB/s	Avg storage consumption bytes/s
SUT1	1.8 %	1,028 kB/s	1,077,844 bytes/s
SUT2	2.5%	1,030 kB/s	1,083,607 bytes/s
SUT3	1.1 %	1,028 kB/s	1,083,607 bytes/s

Table 5.7: *scenario 7 test results*

5.2.3 Scenario 8

In this scenario, we tested all three systems to generate one block every 30 second. The block will contain data with the size of 30MB. Each generated block will be sent to client-nodes. This means that we send a block with data size of 30MB every 30 seconds. The results for test scenario 8 can be seen in 5.8.

System	Avg CPU usage %	Avg bandwidth kB/s	Avg storage consumption bytes/s
SUT1	1.5 %	1,030 kB/s	1,077,844 bytes/s
SUT2	1.92%	1,030 kB/s	1,048,601 bytes/s
SUT3	0.99 %	1,029 kB/s	1,048,601 bytes/s

Table 5.8: *scenario 8 test results*

5.2.4 Scenario 9

In this scenario, we tested all three systems to generate one block each minute. The block will contain data with the size of 60MB. Each generated block will be sent to client-nodes. This means that we send a block with data size of 60MB every minute. The results for test scenario 9 can be seen in 5.9.

System	Avg CPU usage %	Avg bandwidth kB/s	Avg storage consumption bytes/s
SUT1	1.3 %	1,029 kB/s	1,077,844 bytes/s
SUT2	1.7%	1,032 kB/s	1,048,588 bytes/s
SUT3	0.99 %	1,030 kB/s	1,048,588 bytes/s

Table 5.9: *scenario 9 test results*

5.2.5 Overall results for batch mode with data

In each test scenario, each system had almost identical results in term of storage consumption and network bandwidth usage. The only difference was CPU utilization for each system. The following graph 5.2 summaries the results for CPU utilization in streaming mode.

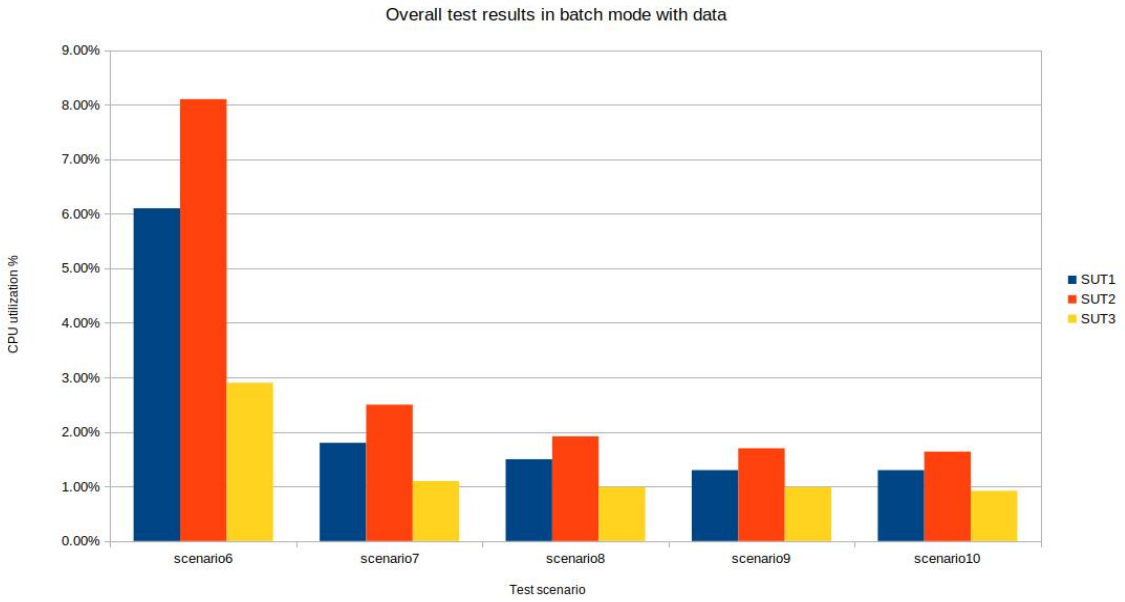

Fig. 5.2: *Overall CPU utilization in batch mode with data*

5.2.6 Scenario 10

In this scenario, we tested all three systems to generate one block every 100 seconds. The block will contain data with the size of 90MB. Each generated block will be sent to client-nodes. This means that we send a block with data size of 90MB every 100 seconds. The results for test scenario 10 can be seen in 5.10.

System	Avg CPU usage %	Avg bandwidth kB/s	Avg storage consumption bytes/s
SUT1	1.3 %	926 kB/s	1,258,301 bytes/s
SUT2	1.64%	927 kB/s	1,258,301 bytes/s
SUT3	0.92 %	928 kB/s	1,258,300 bytes/s

Table 5.10: *scenario 10 test results*

5.3 Batch mode without data

In the following test scenarios (11 -15) we the block without the data. This means that we send a block each second with only information about the actual data.

5.3.1 Scenario 11

In this scenario, we tested all three systems to generate one block each second. The block will not contain any data, but it will contain hashes and timestamps that represents the data. The data will have the size of 1MB. Each generated block will be sent to client-nodes. The results for test scenario 11 can be seen in 5.11.

System	Avg CPU usage %	Avg bandwidth kB/s	Avg storage consumption bytes/s
SUT1	4.99 %	9.9 kB/s	830 bytes/s
SUT2	6.4 %	10 kB/s	832 bytes/s
SUT3	2.6 %	9.8kB/s	828 bytes/s

Table 5.11: *scenario 11 test results*

5.3.2 Scenario 12

In this scenario, we tested all three systems to generate one block every ten seconds. The block will not contain any data, but it will contain hashes and timestamps that represents the data. The data will have the size of 10MB. Each generated block will be sent to client-nodes. The results for test scenario 12 can be seen in 5.12.

System	Avg CPU usage %	Avg bandwidth kB/s	Avg storage consumption bytes/s
SUT1	0.9 %	1.1 kB/s	80 bytes/s
SUT2	1.2 %	1.3 kB/s	76 bytes/s
SUT3	0.4 %	1.5kB/s	76 bytes/s

Table 5.12: *scenario 12 test results*

5.3.3 Scenario 13

In this scenario, we tested all three systems to generate one block every 30 seconds. The block will not contain any data, but it will contain hashes and timestamps that represents the data. The data will have the size of 30MB. Each generated block will be sent to client-nodes. The results for test scenario 13 can be seen in 5.13.

System	Avg CPU usage %	Avg bandwidth kB/s	Avg storage consumption bytes/s
SUT1	0.57 %	0.72 kB/s	25.5 bytes/s
SUT2	0.73 %	0.73kB/s	25.5 bytes/s
SUT3	0.33 %	0.65kB/s	25.3 bytes/s

Table 5.13: *scenario 13 test results*

5.3.4 Scenario 14

In this scenario, we tested all three systems to generate one block every 60 seconds. The block will not contain any data, but it will contain hashes and timestamps that represents the data. The data will have the size of 60MB. Each generated block will be sent to client-nodes. The results for test scenario 14 can be seen in 5.14.

System	Avg CPU usage %	Avg bandwidth kB/s	Avg storage consumption bytes/s
SUT1	0.43 %	0.39 kB/s	12.7 bytes/s
SUT2	0.60 %	0.38 kB/s	12.5 bytes/s
SUT3	0.30 %	0.38 kB/s	12.6 bytes/s

Table 5.14: *scenario 14 test results*

5.3.5 Scenario 15

In this scenario, we tested all three systems to generate one block every 100 seconds. The block will not contain any data, but it will contain hashes and timestamps that represents the data. The data will have the size of 90MB. Each generated block will be sent to client-nodes. The results for test scenario 15 can be seen in 5.15.

System	Avg CPU usage %	Avg bandwidth kB/s	Avg storage consumption bytes/s
SUT1	0.39 %	0.35 kB/s	7.6 bytes/s
SUT2	0.51 %	0.35 kB/s	7.9 bytes/s
SUT3	0.26 %	0.33 kB/s	7.7 bytes/s

Table 5.15: *scenario 15 test results*

5.3.6 Overall results for batch mode without data

In each test scenario, each system had almost identical results in term of storage consumption and network bandwidth usage. The only difference was CPU utilization for each system. The following graph 5.3 summaries the results for CPU utilization in streaming mode.

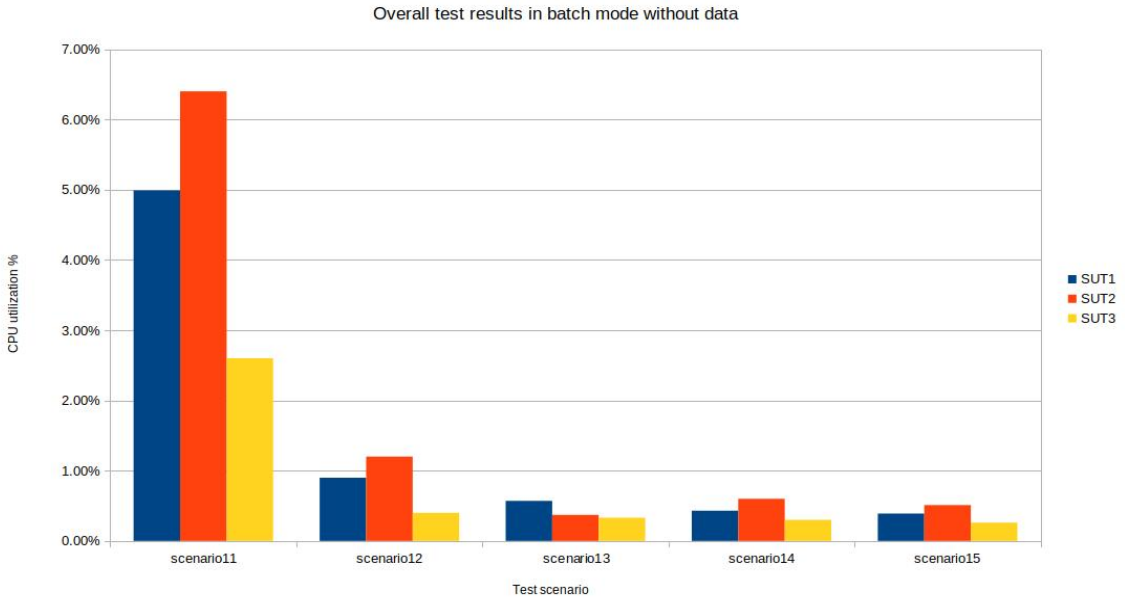

Fig. 5.3: *Overall CPU utilization in batch mode without data*

Chapter 6

Analysis and Discussion

6.1 Blockchain and ethic

As discussed in previous chapters, Blockchain adds layers of security to its users. It works in append mode, which means that data can only be added to the chain. This prohibits anyone from tampering with the data inside the blocks. This is achieved as the chain can be rebuilt, and all hashes can be calculated and verified. This ensures the integrity of the data for the involved parties. By using Blockchain technology, we can be sure that the data will remain untampered.

In 2016, the General Data Protection Regulation (GDPR) was passed by the Council of the European Union. The aim of this regulation is to give the citizens of EU control over their own personal data[18]. This means that any EU citizen can demand any company to delete all his/here relevant data. Blockchain is built around the principle of append-only data. This means that any added data will remain a part of the chain forever. Having said that is recommended to avoid using Blockchain in applications that collect personal data. The intended use of Blockchain is other applications where the data can not be deleted. An example of such application is bank transactions or taxation system. When Blockchain is used is such an application, it will add more security and transparency for both the agency and the client.

6.2 Streaming mode

6.2.1 Scenario 1

In this case we tested all three SUTss to find the maximum number of logs that the system can handel each second in *stream mode*. We expected to have different results based on the the configuration of each SUT. The results shows that the maximum number of blocks is ten blocks/s regardless of system configuration. Although blocks were generated much faster on the master-node (1000 block/s), but only 10 blocks could be delivered each second. The results for storage consumption and network usage came almost identical for the three SUTs. The variation may depend on the moment that monitoring tools took the measurement of system resources. This means that each system will need an average bandwidth $\approx 100kB/s$ dedicated for Blockchain application. In term of storage usage each node will consume about 10kB/s. This means that each block(log) will consume 1kB of hard disk storage. In total each second the system will consume $X * 10kB$, where X is the number of nodes

in the network. In our case where we have one master and one node, it means that the total storage consumption is $2 * 10kB = 20kB/s$.
In term of CPU consumption (compared to SUT1) we found that:

- In SUT2 with 2 cores more and 4 GB of ram more, the CPU load were reduce by about 17%.
- In SUT3 with 6 more cores and 12 GB of ram more, the CPU load were reduced by about 56%.

6.2.2 Scenario 2

In this case we tested all three SUTs to send 5 blocks(logs) each second in *stream mode*. The results for storage consumption and network usage came almost identical for the three SUTs. The variation may depend on the moment that monitoring tools took the measurement of system resources. This means that each system will need an average bandwidth $\approx 50kB/s$ dedicated for Blockchain application. In term of storage usage each node will consume about 5kB/s. In total each second the system will consume $X * 5kB$, where X is the number of nodes in the network. In our case where we have one master and one client, it means that the total storage consumption is $2 * 5kB = 10kB/s$.
In term of CPU consumption (compared to SUT1) we found that:

- In SUT2 with 2 cores more and 4 GB of ram more, the CPU load were increased by about 78%.
- In SUT3 with 6 more cores and 12 GB of ram more, the CPU load were reduced by about 36%.

What is remarkable is that SUT2 had worse results than SUT1 which had 2 cores less and 4 GB of ram less.

6.3 Scenario 3

In this case we tested all three SUTss to send 2 blocks(logs) each second in *stream mode*. The results for storage consumption and network usage came almost identical for the three SUTss. The variation may depend on the moment that monitoring tools took the measurement of system resources. This means that each system will need an average bandwidth $\approx 20kB/s$ dedicated for Blockchain application. In term of storage usage each node will consume about 2kB/s. In total each second the system will consume $X * 2kB$, where X is the number of nodes in the network. In our case where we have one master and one client, it means that the total storage consumption is $2 * 2kB = 4kB/s$.
In term of CPU consumption (compared to SUT1) we found that:

- In SUT2 with 2 cores more and 4 GB of ram more, the CPU load were increased by about 21%.
- In SUT3 with 6 more cores and 12 GB of ram more, the CPU load were reduced by about 58%.

What is remarkable her in that SUT2 had worse results than SUT1 which had 2 cores less and 4 GB of ram less.

6.3.1 Scenario 4

In this case we tested all three SUTss to send 1 blocks(logs) each 0.75 second in *stream mode*. The results for storage consumption and network usage came almost identical for the three SUTss. The variation may depend on the moment that monitoring tools took the measurement of system resources. This means that each system will need an average bandwidth $\approx 13kB/s$ dedicated for Blockchain application. In term of storage usage each node will consume about 1.4kB/s. In total each second the system will consume $X * 1.4kB$, where X is the number of nodes in the network. In our case where we have one master and one client, it means that the total storage consumption is $2 * 1.4kB = 2.8kB/s$.
In term of CPU consumption (compared to SUT1) we found that:

- In SUT2 with 2 cores more and 4 GB of ram more, the CPU load were increased by about 21%.
- In SUT3 with 6 more cores and 12 GB of ram more, the CPU load were reduced by about 60%.

What is remarkable her in that SUT2 had worse results than SUT1 which had 2 cores less and 4 GB of ram less.

6.3.2 Scenario 5

In this case we tested all three SUTss to send 1 blocks(logs) each second in *stream mode*. The results for storage consumption and network usage came almost identical for the three SUTss. The variation may depend on the moment that monitoring tools took the measurement of system resources. This means that each system will need an average bandwidth $\approx 10kB/s$ dedicated for Blockchain application. In term of storage usage each node will consume about 1.3kB/s. In total each second the system will consume $X * 1.3kB$, where X is the number of nodes in the network. In our case where we have one master and one client, it means that the total storage consumption is $2 * 1.3kB = 2.6kB/s$.
In term of CPU consumption (compared to SUT1) we found that:

- In SUT2 with 2 cores more and 4 GB of ram more, the CPU load were increased by about 6%.

- In SUT3 with 6 more cores and 12 GB of ram more, the CPU load were reduced by about 54%.

What is remarkable her in that SUT2 had worse results than SUT1 which had 2 cores less and 4 GB of ram less.

6.3.3 Overall results for streaming mode

The results for streaming mode shows that the system does not scale up by adding more CPU cores. In scenario 1 we tested each system to find the maximum number of blocks that we could deliver and verify. The maximum number of blocks is 10 for all the system. It is expected to have better performance and more blocks when adding more cores. The results showed that our expectation is not true. By isolation each component of the framework, we were able to identify network communication to be the bottleneck. If network communication is excluded, each system could generate more than 500 blocks each second. Our explanation is that the communication process is done in 4 steps, and the process is repeated for each block. Another reason is that Blockchain works in a serialized manner. This means that we must wait until the client-node receives the block and verify it, before sending a new block. The reason for that is that the previous block in needed to verify the new block.

In the results, we can see that SUT1 works better than SUT2 in almost every case. Here we can not explain the reason for that, and this opens for more future researches to understand this behaviour. But our assumption that more cores result in better performance is true in case of SUT1 and SUT2

6.4 Batch mode with data

6.4.1 Scenario 6

In this case we tested all three SUTss to send data with the size of about 80GB/day in batch mode. The data is sent as data chunks with the size of 1MB each second. The results for storage consumption and network usage came almost identical for the three SUTss. The variation may depend on the moment that monitoring tools took the measurement of system resources. This means that each system will need an average bandwidth $\approx 1.04MB/s$ dedicated for Blockchain application. In term of storage usage each node will consume about 1MB/s. In total each second the system will consume $X * 1MB$, where X is the number of nodes in the network. In our case where we have one master and one client, it means that the total storage consumption is $2 * 1MB = 2MB/s$.
In term of CPU consumption (compared to SUT1) we found that:

- In SUT2 with 2 cores more and 4 GB of ram more, the CPU load were increased by about 33%.
- In SUT3 with 6 more cores and 12 GB of ram more, the CPU load were reduced by about 52%.

What is remarkable her in that SUT2 had worse results than SUT1 which had 2 cores less and 4 GB of ram less.

6.4.2 Scenario 7

In this case we tested all three SUTss to send data with the size of about 80GB/day in batch mode. The data is sent as data chunks with the size of 10MB every 10 second. The results for storage consumption and network usage came almost identical for the three SUTss. The variation may depend on the moment that monitoring tools took the measurement of system resources. This means that each system will need an average bandwidth $\approx 1.03MB/s$ dedicated for Blockchain application. In term of storage usage each node will consume about 1MB/s. In total each second the system will consume $X * 1MB$, where X is the number of nodes in the network. In our case where we have one master and one client, it means that the total storage consumption is $2 * 1MB = 2MB/s$.
In term of CPU consumption (compared to SUT1) we found that:

- In SUT2 with 2 cores more and 4 GB of ram more, the CPU load were increased by about 39%.
- In SUT3 with 6 more cores and 12 GB of ram more, the CPU load were reduced by about 39%.

What is remarkable her in that SUT2 had worse results than SUT1 which had 2 cores less and 4 GB of ram less.

6.4.3 Scenario 8

In this case we tested all three SUTss to send data with the size of about 80GB/day in batch mode. The data is sent as data chunks with the size of 30MB every 30 second. The results for storage consumption and network usage came almost identical for the three SUTss. The variation may depend on the moment that monitoring tools took the measurement of system resources. This means that each system will need an average bandwidth $\approx 1.03MB/s$ dedicated for Blockchain application. In term of storage usage each node will consume about 1MB/s. In total each second the system will consume $X * 1MB$, where X is the number of nodes in the network. In our case where we have one master and one client, it means that the total storage consumption is $2 * 1MB = 2MB/s$.
In term of CPU consumption (compared to SUT1) we found that:

- In SUT2 with 2 cores more and 4 GB of ram more, the CPU load were increased by about 28%.
- In SUT3 with 6 more cores and 12 GB of ram more, the CPU load were reduced by about 34%.

What is remarkable her in that SUT2 had worse results than SUT1 which had 2 cores less and 4 GB of ram less.

6.4.4 Scenario 9

In this case we tested all three SUTss to send data with the size of about 80GB/day in batch mode. The data is sent as data chunks with the size of 60MB every 60 second. The results for storage consumption and network usage came almost identical for the three SUTss. The variation may depend on the moment that monitoring tools took the measurement of system resources. This means that each system will need an average bandwidth $\approx 1MB/s$ dedicated for Blockchain application. In term of storage usage each node will consume about 1MB/s. In total each second the system will consume $X * 1MB$, where X is the number of nodes in the network. In our case where we have one master and one client, it means that the total storage consumption is $2 * 1MB = 2MB/s$.
In term of CPU consumption (compared to SUT1) we found that:

- In SUT2 with 2 cores more and 4 GB of ram more, the CPU load were increased by about 31%.
- In SUT3 with 6 more cores and 12 GB of ram more, the CPU load were reduced by about 24%.

What is remarkable her in that SUT2 had worse results than SUT1 which had 2 cores less and 4 GB of ram less.

6.4.5 Scenario 10

In this case we tested all three SUTss to send data with the size of about 77 GB/day in batch mode. The data is sent as data chunks with the size of 90MB every 100 second. The results for storage consumption and network usage came almost identical for the three SUTss. The variation may depend on the moment that monitoring tools took the measurement of system resources. This means that each system will need an average bandwidth $\approx 0.9MB/s$ dedicated for Blockchain application. In term of storage usage each node will consume about 1.3MB/s. In total each second the system will consume $X * 1.3MB$, where X is the number of nodes in the network. In our case where we have one master and one client, it means that the total storage consumption is $2 * 1.3MB = 2.6MB/s$.
In term of CPU consumption (compared to SUT1) we found that:

- In SUT2 with 2 cores more and 4 GB of ram more, the CPU load were increased by about 26%.
- In SUT3 with 6 more cores and 12 GB of ram more, the CPU load were reduced by about 29%.

What is remarkable her in that SUT2 had worse results than SUT1 which had 2 cores less and 4 GB of ram less.

6.4.6 Overall results for batch mode with data

In terms of average bandwidth usage and storage consumption, all the systems in all test scenarios (6-10) are the same. The reason for is that in every test case, we send the same amount of data, and this results in having the same average bandwidth usage and storage consumption. Only different here is CPU consumption. The reason for that is, reducing the total number of blocks each second will result in less CPU consumption and fewer network connections(new connection on every block).

6.5 Batch mode without data

6.5.1 Scenario 11

In this case we tested all three SUTs to send a block each second in batch mode. The block will not contain any data, but it will contain hashes and timestamps that represents the data. Here the data which the block holds information about, has the size of 1MB. The results for storage consumption and network usage came almost identical for the three SUTs. The variation may depend on the moment that monitoring tools took the measurement of system resources. This means that each system will need an average bandwidth $\approx 10KB/s$ dedicated for Blockchain application. In term of storage usage each node will consume about 1kB/s. In total each second the system will consume $X * 1kB$, where X is the number of nodes in the network. In our case where we have one master and one client, it means that the total storage consumption is $2 * 1kB = 2kB/s$.
In term of CPU consumption (compared to SUT1) we found that:

- In SUT2 with 2 cores more and 4 GB of ram more, the CPU load were increased by about 28%.
- In SUT3 with 6 more cores and 12 GB of ram more, the CPU load were reduced by about 48%.

What is remarkable her in that SUT2 had worse results than SUT1 which had 2 cores less and 4 GB of ram less.

6.5.2 Scenario 12

In this case we tested all three SUTs to send a block every 10 second in batch mode. The block will not contain any data, but it will contain hashes and timestamps that represents the data. Here the data which the block holds information about, has the size of 10MB. The results for storage consumption and network usage came almost identical for the three SUTs. The variation may depend on the moment that monitoring tools took the measurement of system resources. This means that each system will need an average bandwidth $\approx 1.3kB/s$ dedicated for Blockchain application. In term of storage usage each node will consume about 80 Byte/s. In total each second the system will consume $X * 80Byte$, where X is the number of nodes in the network. In our case where we have one master and one client, it means that the total storage

consumption is $2 * 80Byte = 160Byte/s$.
In term of CPU consumption (compared to SUT1) we found that:

- In SUT2 with 2 cores more and 4 GB of ram more, the CPU load were increased by about 33%.
- In SUT3 with 6 more cores and 12 GB of ram more, the CPU load were reduced by about 56%.

What is remarkable her in that SUT2 had worse results than SUT1 which had 2 cores less and 4 GB of ram less.

6.5.3 Scenario 13

In this case we tested all three SUTs to send a block every 30 second in batch mode. The block will not contain any data, but it will contain hashes and timestamps that represents the data. Here the data which the block holds information about, has the size of 30MB. The results for storage consumption and network usage came almost identical for the three SUTs. The variation may depend on the moment that monitoring tools took the measurement of system resources. This means that each system will need an average bandwidth $\approx 700Bytes/s$ dedicated for Blockchain application. In term of storage usage each node will consume about 25 Byte/s. In total each second the system will consume $X * 25Byte$, where X is the number of nodes in the network. In our case where we have one master and one client, it means that the total storage consumption is $2 * 25Byte = 50Byte/s$.
In term of CPU consumption (compared to SUT1) we found that:

- In SUT2 with 2 cores more and 4 GB of ram more, the CPU load were increased by about 28%.
- In SUT3 with 6 more cores and 12 GB of ram more, the CPU load were reduced by about 42%.

What is remarkable her in that SUT2 had worse results than SUT1 which had 2 cores less and 4 GB of ram less.

6.5.4 Scenario 14

In this case we tested all three SUTs to send a block every 60 second in batch mode. The block will not contain any data, but it will contain hashes and timestamps that represents the data. Here the data which the block holds information about, has the size of 60MB. The results for storage consumption and network usage came almost identical for the three SUTs. The variation may depend on the moment that monitoring tools took the measurement of system resources. This means that each system will need an average bandwidth $\approx 400Bytes/s$ dedicated for Blockchain application. In term of storage usage each node will consume about 12.5 Byte/s. In total each second the system will consume $X * 12.5Byte$, where X is the number of nodes in

the network. In our case where we have one master and one client, it means that the total storage consumption is $2 * 12.5Byte = 25Byte/s$.
In term of CPU consumption (compared to SUT1) we found that:

- In SUT2 with 2 cores more and 4 GB of ram more, the CPU load were increased by about 40%.
- In SUT3 with 6 more cores and 12 GB of ram more, the CPU load were reduced by about 30%.

What is remarkable her in that SUT2 had worse results than SUT1 which had 2 cores less and 4 GB of ram less.

6.5.5 Scenario 15

In this case we tested all three SUTs to send a block every 100 second in batch mode. The block will not contain any data, but it will contain hashes and timestamps that represents the data. Here the data which the block holds information about, has the size of 90MB. The results for storage consumption and network usage came almost identical for the three SUTs. The variation may depend on the moment that monitoring tools took the measurement of system resources. This means that each system will need an average bandwidth $\approx 350Bytes/s$ dedicated for Blockchain application. In term of storage usage each node will consume about 7.7 Byte/s. In total each second the system will consume $X * 7.7Byte$, where X is the number of nodes in the network. In our case where we have one master and one client, it means that the total storage consumption is $2 * 7.7Byte = 15.4Byte/s$.
In term of CPU consumption (compared to SUT1) we found that:

- In SUT2 with 2 cores more and 4 GB of ram more, the CPU load were increased by about 31%.
- In SUT3 with 6 more cores and 12 GB of ram more, the CPU load were reduced by about 33%.

What is remarkable her in that SUT2 had worse results than SUT1 which had 2 cores less and 4 GB of ram less.

6.5.6 Overall results for batch mode with data

In test scenarios, 11 to 15, the average bandwidth usage, CPU utilization and storage consumption are different on each case. In case of storage consumption, we use less storage as we extend the time between the blocks. By sending a block every 60 seconds, we reduce the number of blocks from 60 to one block (compared to sending a block each second). In both cases, we send the same amount of data, but the different her is the number of blocks. By sending an empty block with only the hashes, we reduce the total storage consumption. This will result in reducing the average bandwidth usage and CPU utilization as we send less data.

Chapter 7

Conclusions and Future Work

7.1 Can Blockchain be used as evidence?

The code of judicial procedure is based on the principle of free evidence. It has not been adapted to conform to any type of evidence. This means that any evidence, regardless of the form or the type, can be used and presented in a Swedish court. Only the judge can decide what is to be used and what to be excluded. Our conclusion here is that Blockchain can be used as digital evidence in Swedish court as the principle of free evidence states that any form of evidence can be presented.

7.2 Computer resource usage in stream mode

In our tests we found that the bottleneck is network. The maximum number of blocks that the system can manage is ten blocks/s. Adding more cores to our SUT did not increase the number of blocks. In tests, we found that service time for for the system is 100ms. This means that the time elapsed from the moment a new block is created until it is sent to the clients is 100ms. In our tests, we found that we could generate more than 500 blocks/s by excluding the time for notifying the clients and sending the blocks. In term of storage usage, we found that each block is at least five times larger than the original log stream. This extra storage is introduced as we save additional data in the form of hashes and timestamps to ensure the integrity of the logs. When it comes to CPU consumption, our tests show that the system does not benefit from extra cores. The results were exactly the opposite in the case of SUT2; the system showed higher CPU consumption. In the case of SUT3, the system had lower CPU consumption. Our conclusion here is that SUT1 with two cores and 4GB of memory and SUT3 with eight cores and 16 GB of memory works best in our tests. The system does not scale in term of the maximum number of blocks/s as we are limited to ten blocks/s

7.3 Computer resource usage in batch mode

In batch mode, we tested to send about 80GB of data each day. We tested to divide the data into different sizes, and we tested to have different time periods between two blocks. The time between two blocks varied between 1, 10, 30, 60 and 100 seconds. In all tests, CPU consumption was under 10% of the total CPU utilization. Here, we also found that SUT1 with two cores and 4GB of memory and SUT3 with eight

cores and 16 GB of memory works best in our tests. Testing to send the blocks without including the data showed an improvement in the CPU consumption. The most important thing to notice here is the reduction in storage usage. Here each node will store the blocks without the data. Here we win in term of storage as the size of the blocks become negligible when it is compared to the total data size. Our conclusion here We can save data storage if we send blocks without the data. When it comes to CPU consumption, Sending larger files in a longer period of time between each block will reduce the consumption of the CPU power.

7.4 Future work

Here we will demonstrate what we think can be improved in our implementation to achieve higher performance and maximize the number of blocks created each second.

7.4.1 Network optimization

As said before we found out that Network is the bottleneck in our implementation. We suggest more study on how to improve and reduce service time. The number of blocks that can be generated is higher than the number of blocks that we can send to the client nodes. As for now, we create a new socket connection and a new authentication process for each block. We believe that we can increase the number of sent blocks by allowing multiple blocks to be sent in the same session. Another way to achieve that is to set each session to a fixed time x. By that, we mean that every generated block in x seconds will be sent in the same session. After that, the session will be terminated, and a new session will be created

7.4.2 Real-world test

As for now, we do not see any obstacle that prevents us from using Blockchain as evidence. Here we suggest using Blockchain to hold log data. In case of a dispute we recommend to use Blockchain as evidence in court. By using it in a court, we can test our Blockchain on a real case and find out if it can be accepted as evidence.

All tests in our study were conducted in lab environment. We suggest to test the frame work in a real-world application. We also suggest testing the framework with multiple client nodes. A possible application for the system is to integrate it with the banking system. In this way we can grant the integrity of all transactions and clients data. As well as we can grant that the data will be secure against manipulation.

References

[1] Block size limit. [Online]. https://bitcoin.org/en/glossary/block-size-limit. [Accessed : 05 - Feb- 2019].

[2] Case study - blockchain in hyperledger: Better than etl? [Online]. https://keyholesoftware.com/company/creations/tech-resources/hyperledgerblockchain-casestudy/. [Accessed : 06 - Feb- 2019].

[3] Fri bevisprövning. [Online]. https://www.aklagare.se/ordlista/f/fri-bevisprovning/. [Accessed : 10 - Feb- 2019].

[4] Polisen slog till mot the pirate bay, 2006. [Online]. https://www.realtid.se/polisen-slog-till-mot-pirate-bay. [Accessed : 01 - Feb- 2019].

[5] Inga skadestånd till drabbade i piratrazzia, 2007. [Online]. https://computersweden.idg.se/2.2683/1.113812/inga-skadestand-till-drabbade-i-piratrazzia. [Accessed : 01 - Feb- 2019].

[6] The swedish code of judicial procedure, 2015. [Online]. https://www.government.se/government-policy/judicial-system/the-swedish-code-of-judicial-procedure/?fbclid=IwAR2MjInX9VfdC7r1UnTGQPtpNRTXitjYJykMSXTA39KEyh08pOg2Xevw9k0. [Accessed : 10 - Feb- 2019].

[7] Chinmay Ektare Deepti Thomas Syed Akram Akanksha Kaushik, Archana Choudhary. Blockchain — literature survey. *Published in: 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology*, 2, 2018.

[8] Imran Bashir. *Mastering Blockchain*. Packt Publishing, 2^{ed} edition, 2018.

[9] Nolan Bauerle. What is blockchain technology., 2018. [Online]. https://www.coindesk.com/information/what-is-blockchain-technology. [Accessed : 01 - Dec- 2018].

[10] Manion Lawrence Cohen Louis and Morrisson Keith. *Research methods in education*. Routledge, 7th edition, 2011.

[11] Laura Marissa Cullell. Digital forensics and blockchain, 2018. [Online]. https://medium.com/@blockxlabs/digital-forensics-and-blockchain-bf3af5e7153c. [Accessed : 01 - Dec- 2018].

[12] Magnus Hellberg Erik Högström. Här slår polisen till – mot the pirate bay, 2014. [Online]. https://www.expressen.se/nyheter/har-slar-polisen-till--mot-the-pirate-bay/. [Accessed : 01 - Feb- 2019].

[13] Mcniff Jean and Whitehead Jack. *All you need to know about Action Research*. SAGE publication Ltd, 2nd edition, 2012.

[14] Giovanni KESSLER. Guidelines on digital forensic procedures for olaf staff, 2016. [Online]. Available: Official website of the European Union, https://europa.eu. [Accessed : 01 - Dec- 2018].

[15] S Kronqvist. *Brott och digitala bevis*. Norstedts Juridik, 3rd edition, 2013.

[16] Auqib Hamid Lone. Forensic-chain: Ethereum blockchain based digital forensics chain of custody, 2017. [Online]. Available: Gesearch Gate, https://www.researchgate.net. [Accessed : 01 - Dec- 2018].

[17] Oliver Hinz Dirk Schiereck Michael Nofer, Peter Gomber. Blockchain. *Bus Inf Syst Eng*, 7:183–187, 2017.

[18] Council of European Union. Council regulation (eu) no 2016/679., 2016. [Online]. Available: Council of European Union, https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2016.119.01.0001.01.ENG. [Accessed : 10 - april- 2019].

[19] Norwich University Online. 5 steps for conducting computer forensics investigations, 2017. [Online]. https://online.norwich.edu/academic-programs/resources/5-steps-for-conducting-computer-forensics-investigations. [Accessed : 01 - Dec- 2018].

[20] Park J.Y. Park J.H. and Huh E.N. Block chain based data logging and integrity management system for cloud forensic. *ICCSEA 2017*, 7:149–159, 2017.

[21] Chetan Naik Manjunath KN Srikanth Prabhu PSaurabh Dhumwad, Mandar Sukhadeve. A peer to peer money transfer using sha256 and merkle tree. *ADCOM 2017*, 2017.

[22] Koshik Raj. *Foundations of Blockchain*. Packt Publishing, 1st edition, 2019.

[23] Kumar Ranjit. *Research methodology a step-by-step guied for beginners*. SAGE publication Ltd, 4th edition, 2014.

[24] Alfonso de la Rocha. Is someone bothering about performance evaluation in blockchain systems?, 2018. [Online]. https://medium.com/@adlrocha/is-someone-bothering-about-performance-evaluation-in-blockchain-systems-58b04aa3dc69. [Accessed : 01 - Dec- 2018].

[25] Mattias Scherer. *Performance and Scalability of Blockchain Networks and Smart Contracts*. 2017.

[26] Justin Szilard. Google set to revolutionize blockchain search-ability, sheds new light on cryptocurrency uses.

[27] Wei Yu Weichao Gao, William G. Hatcher. Blockchain challenges and opportunities: a survey. *International Conference on Computer Communication and Networks*, 27, 2018.

[28] Int. J. Z. Zheng et al. Blockchain challenges and opportunities: a survey. *Web and Grid Services*, 20, 2018.

[29] Christian Zoubek and Konstantin Sack. Selective deletion of non-relevant data. *Digital Investigation*, 20:92–98, 2017.

Appendix A

Supplemental Information

A.0.1 Collected data representation for SUT1 in scenario 1

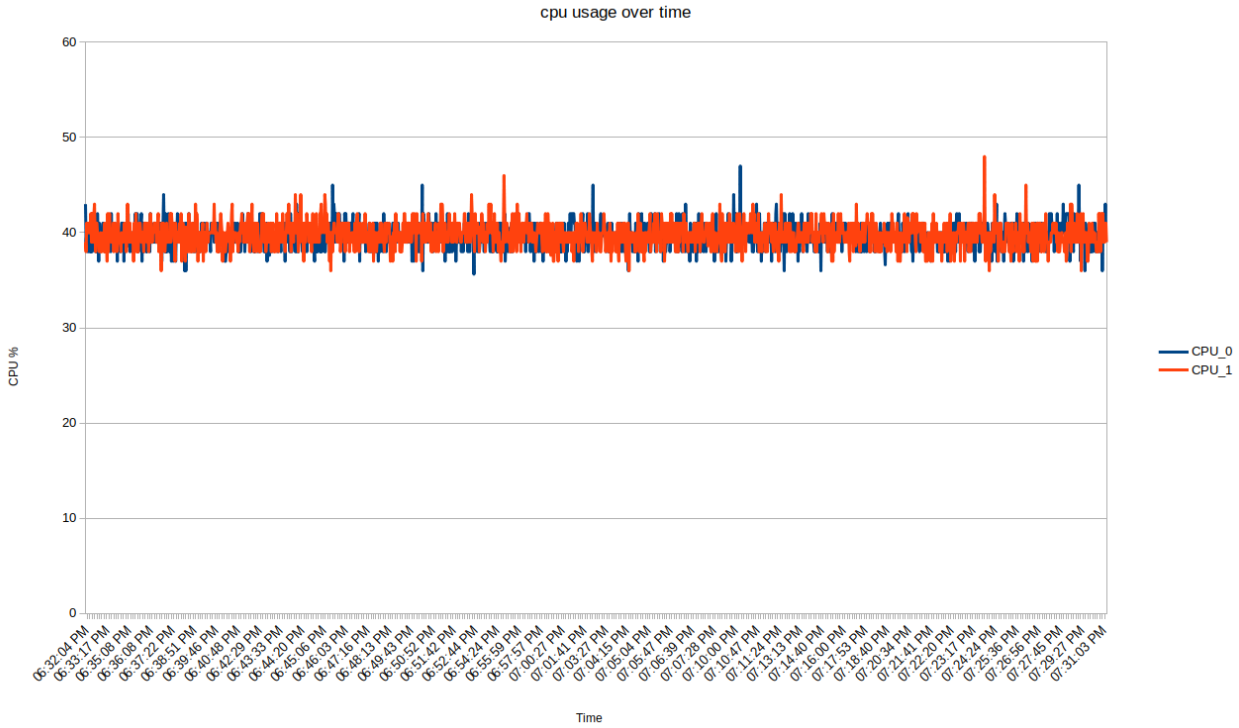

Fig. A.1: *CPU utilization For SUT1 in scenario1*

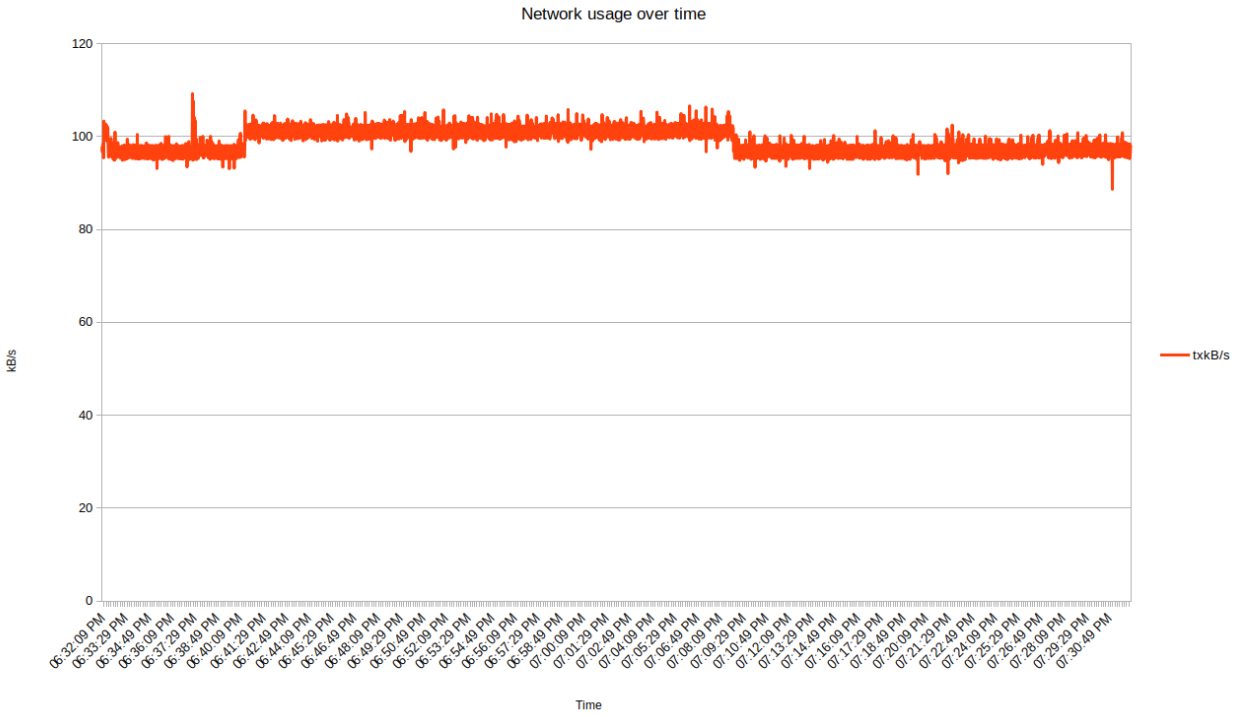

Fig. A.2: *Network utilization For SUT1 in scenario1*

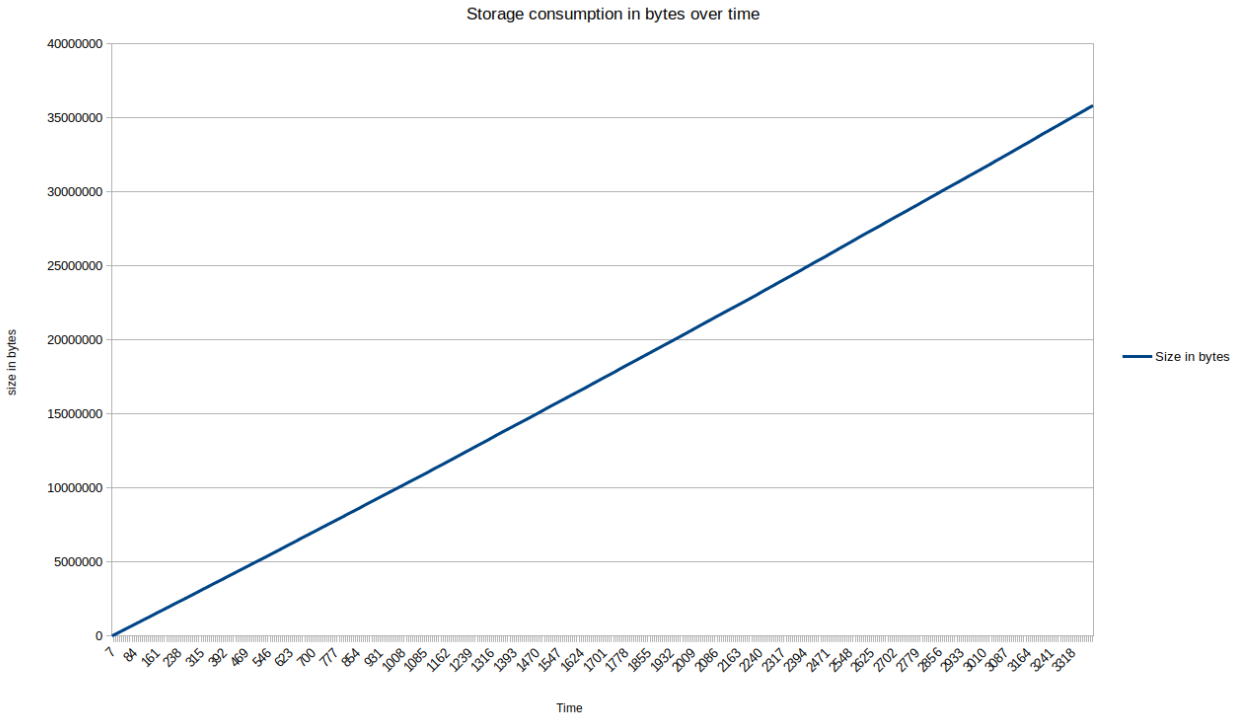

Fig. A.3: *Storage consumption For SUT1 in scenario1*

Faculty of Computing, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden