Master Security and Network Engineering

University of Amsterdam

Using a Verifiable and Decentralized Ledger as a Basis for Trusting Hospital Endpoints

Matthijs Bartelink

July 4, 2021

Supervisor(s): Guido van't Noordende

Signed:

Abstract

The whitebox system is a system for controlling access to confidential medical data. General practicioners hold confidential data for their patients, and own a whitebox which controls remote access to that data. These whiteboxes establish hospital endpoints as trusted to access confidential data by having patients transfer a code to the hospital organization. Once enough of these code transfers have taken place, an endpoint is considered trusted.

We propose a system for registering trust between an endpoint and a whitebox. By recording trust by whiteboxes in a decentralized blockchain ledger, we allow whiteboxes to check trust in an endpoint by other whiteboxes. Endpoints trusted by other whiteboxes may be considered trusted. The ledger uses a proof-of-authority consensus algorithm, where whiteboxes are the authority on blocks describing their own trust. By recording both positive and negative trust, the system provides a mechanism for removing the trust from an endpoint that acts maliciously.

We analyze the security of the proposed system by considering its vulnerability to denial-of-service and inference attacks, as well as attempts to establish false trust by malicious whiteboxes. Finally, we analyze the scaling of a prototype implementation. For application within the dutch healthcare system, we find that the proposed system scales well enough both in term of disk-space use and trust check speed.

Contents

CHAPTER 1

Introduction

Whitebox Systems is a company which produces a decentralized system for controlling access to patient data. The Whitebox itself is a computer in the possession of the *general practitioner (GP)* which controls access to patient files stored on the *healthcare information system (HIS)*. If a party requires access to patient data, it should obtain an authorization URL first; the source system (GP/HIS) takes the initiative to send a URL to a recipient as a form of authorization. The use of these URLs is then tracked and combined with UZI-numbers of the doctors accessing the files.

This system currently has two methods for determining who can access patient data. The first method is simply to configure the Whitebox to trust certain systems of other healthcare professionals by having the GP configure in an address of this system in the Whitebox. Over this "trust-link" authorization URL's can be sent when required. The second method is to establish trust via having the patient transfer a code to the healthcare organization which seeks access to the data, which together with the citizen identification number (BSN) of the patient can be used to find the URL containing the patients information. In this latter scenario, the authorized party is an individual doctor – who has to use a unique personal healthcare professional (UZI) smartcard – not an organization.

Untrusted endpoints of system components (e.g., for identity management [4]) of healthcare organizations, which are authenticated by the second method, can become trusted by having multiple different doctors request patient data using the second method. When using an authorization code to open a patient file, the doctor can endorse the endpoints and, for example, the person who signs information of an organization. These endorsements are stored, and a system is considered trusted once it has a certain number of them. It is this mechanism this project will seek to explore.

Currently each Whitebox decides who to trust separately, with each new Whitebox repeating the process of establishing trust with healthcare organizations requesting data. We propose a blockchain based ledger of established trust which can be used to identify trustworthy healthcare organization addresses. A healthcare organization could then establish trust by relying on trust already established by other Whiteboxes.

1.1 Research question and approach

The central goal of this project is to use a verifiable ledger of trust by GPs as a decentralized method of identifying trusted hospital endpoint addresses; such endpoints may, for example, contain (certified/signed) information on the doctors who work in a hospital and their credentials, or be used to send authorization-URLs to.

In essence, the trust fabric proposed in this project helps (re)enforce confidence in the authenticity of information stored in resources of different hospitals. We break this down into the following subquestions.

How can we store established trust relations in a verifiable way?

How can we allow for revocation when trust is stored this way?

How can we use a ledger of trust relations as a method for deciding wether or not a whitebox is trusted?

How can we account for the loss and revocation of trust in this system?

How well does this system scale for its intended use?

1.2 Approach Outline

We start by examining the context of the research, starting by examining similar projects by others and developing an understanding of the whitebox system. From there we propose a system to analyze the trust which functions as extension of the whitebox system. We outline the requirements for the proposed system, then outline a design to meet those requirements. A prototype is implemented using Python, Flask and SQLite. This prototype is then used as a proof of concept for the design, and to analyze the scaling of the proposed system.

CHAPTER 2

Background and related work

2.1 Related Work

Bubbles of Trust: a decentralized Blockchain-based authenticationsystem for IoT

This paper by Mohamed Tahar Hammi, Badis Hammi, Patrick Bellot and Ahmed Serhrouchni discussed a decentralized system for the identification and authentication of IOT devices. The system discussed cannot be repurposed here, as it doesn't allow new additions after it's establishment. The threat model used is very solid however, and is an inspiration for the threat model used in this research project.

A framework for secure and decentralized sharing of medical imaging data via blockchain consensus

In this paper Vishal Patel discusses using a blockchain based log as a way of managing who can view medical imaging data based on patient consent. The idea being to record radiological studies, URL's where their images can be found, and patient records to consent to viewing within the same blockchain, to construct a complete record of the studies performed and who may view their results.

Since the goal of the blockchain is very different, the ideas in this paper cannot be directly repurposed. It does provide a useful explanation of blockchain in general and goes into the possible ethical issues with using it for medical purposes.

Bitcoin: A Peer-to-Peer Electronic Cash System

This is the paper that originally introduced bitcoin. It explains blockchain, though without using that name, as a method to prevent double spending for a currency of cryptography hashes. Since we are looking to repurpose the idea of storing transactions in a verifiable way using a blockchain, it remains a valuable reference.

Identiteit en authenticatie: onze zorg(dutch: Identity and authentication: our healthcare

This report, produced by Guido van't Noordende and Bas Kloosterman from Whitebox Systems, gives an overview of the identification and authorization techniques currently used and proposed in dutch healthcare. It is a great reference for understanding the context of this research.

2.2 The whitebox system

Whiteboxes are communication systems used for the exchange of confidential patient data controlled by a GP. By having a physical device owned and controlled by the GP, the GP is allowed direct control over which outside organizations have access to confidential data. The system prevents the spread of patient data by having healthcare organizations reacquire copies of patient files for repeated viewing rather than storing a local copy. Only authorized parties get access to patient data. This is a considerable advantage over conventional centralized exchange systems, where all participants in the system get access to all confidential data shared via that system.

CHAPTER 3

Design

3.1 Requirements

The idea behind the proposed system is to establish trust in an endpoint based on trust already established in that endpoint by other whiteboxes. Establishing that an endpoint is trusted must take no longer than a few seconds, since the user must wait on this operation. Creating a trust-link to an endpoint or revoking that trust-link may take longer if necessary. The proposed system must be reliably capable of establishing trust status of an endpoint at acceptable speed. It should be resistant to denial of service attacks, remaining accessible as best as possible. Finally, the system must not reveal confidential information about the patients.

The proposed system should be able to scale up to the size of the dutch healthcare system. This would mean a little under 5000 whiteboxes. The number of endpoints is harder to estimate. Counting generously, there are a little under 250 hospitals in the Netherlands. However, one hospital is not necessarily limited to one endpoint. Should we wish to include other healthcare organizations, such as pharmacies or physical therapists, the number increases even more. Luckily, not every whitebox needs to endorse every endpoint as only the endpoints that they interact with need to be endorsed. This allows us to reduce the number of endpoints necessary for a whitebox to support to about 150 or so.

When verifying that an endpoint is considered trusted, the system should give a quick overview of the trust established by other whiteboxes. When considering previously established trust, a decision that an endpoint is not trustworthy is more important than the opposite, and meaningfully distinct from simply no longer having a trust-link with a given endpoint. Therefore, we must store negative trust-links as well. When verifying trust, an overview should be provided that includes the number of trust-links, the number of negative trust-links, and an overview of the age of those trust-links.

3.2 Design Overview

The general idea behind the proposed system is to bootstrap trust between a whitebox and an endpoint by relying on previously established trust-links between that endpoint and other whiteboxes. Each whitebox keeps a ledger for each endpoint it trusts. In each ledger we store operations on the trust-links whiteboxes have with the endpoint of the ledger, the nature of those trust-links(positive and negative) and the age of those trust-links. The contents of these ledgers are used as the basis for trusting the endpoint. A blockchain is used to verify the integrity of the ledger. For the details of the working of blockchain, we will defer to the paper where it was first described [3]. The contents of the ledgers are kept identical across whiteboxes with a trust-link to the same endpoint, with a single ledger tied to an endpoint being copied accross all whiteboxes with a trust-link with that particular endpoint. This means the integrity of a

novel ledger can be verified by asking multiple whiteboxes and comparing hashes of their final blocks. If the hashes are identical, the ledgers must be as well. We use a proof of authority consensus algorithm, with whiteboxes themselves being the valid authorities for their own trust-links.

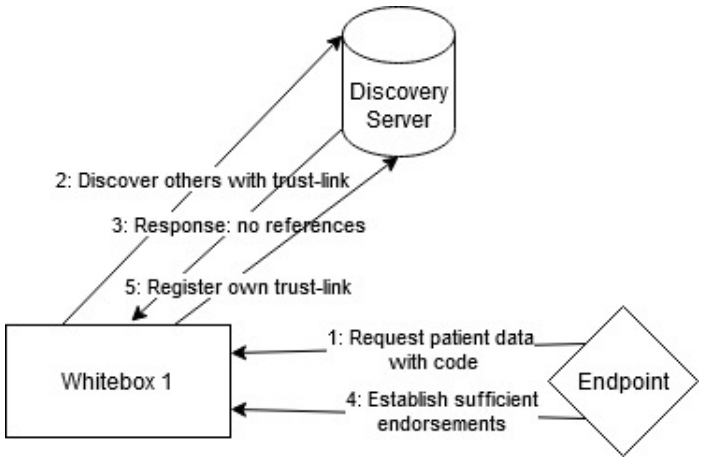

Figure 3.1: A new endpoint tries to access patient data. The whitebox has no references for it, so trust has to be established using an alternate mechanism.

In figure 3.1 we see a previously unknown endpoint contact a whitebox with a request for patient data. The system doesn't actually provide any trust here. The initial request is rejected, and it is only when trust is established via other means(in this case endorsement backed by UZI-numbers) that a trust-link between the whitebox and the endpoint can be established.

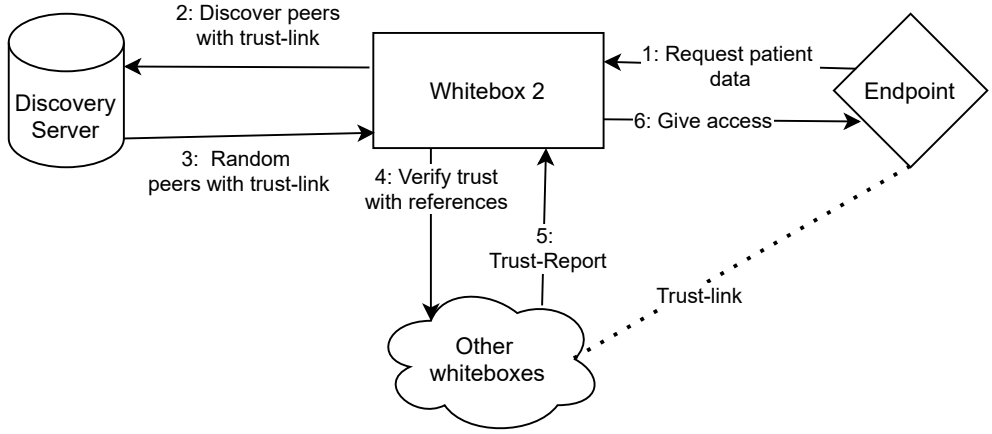

Figure 3.2: A known endpoint tries to access patient data. The whitebox does get references, and uses them to verify trust.

In figure 3.2 we see how a whitebox can use the previously established trust-links from other whiteboxes. In this scenario, trust-links already exist. Therefore, references are acquired when requested from the discovery server. The whitebox verifies the accuracy of these references. If there are a sufficient number of working references, the endpoint is considered trustworthy. Whitebox 2 should establish its own trust-link now.

3.2.1 Previously Established Keys

The proof of authority consensus algorithm proposed depends on the ability to verify the identity of that authority, in this case the whiteboxes. This currently(see section 5.2: future work) requires a centralized source of trust. We propose a key stored on each whitebox, signed centrally and in a verifiable way by the distributor of the whiteboxes. This does then make that distributor a centralized source of trust, which is still relied upon.

Similarly the proposed system assumes the accuracy of the first few trust-links established with an endpoint. Unlike later trust-links, which are based on earlier trust-links, these first few are based on authorized access via another method. That being the exchange of a code between the patient and the healthcare organization which owns the endpoint.

By knowing this code and the patients BSN, the doctor at the healthcare organization can access patient data. At that time, the doctor also has the opportunity to endorse the endpoint with their unique UZI-number. A record of the endorsement is then associated with the endpoint, and a trust-link is established after a certain number of endorsements have been made. If an attacker can fake UZI-number backed endorsements, the proposed system is vulnerable.

3.2.2 Discovery Server

The whiteboxes need some way of finding out where the ledger for a given endpoint already exists. Therefore, some outside entity needs to keep records that they can use to discover other whiteboxes with a trust-link to the given endpoint. This is the purpose of the discovery server. When a whitebox seeks to establish the trustworthiness of an endpoint it doesn't yet know, it can discover peers that already know the endpoint by requesting them from the discovery server. The discovery server always chooses the whiteboxes it responds with at random.

Since discovery can only be done via the discovery server, the discovery server going down would prevent the establishment of new trust-links using the system, as well as any whitebox establishing initial trust by consulting the ledgers of others. Therefore it is strongly recommended to have at least one backup of the discovery server, keeping identical records. Regardless, the discovery server is a central point of failure in the proposed system.

3.2.3 Security Parameter S

We propose a security parameter S, to be used both for the number of whiteboxes queried when checking trust as well as the number of whiteboxes used to verify the integrity of a new ledger. This parameter equals the number of whiteboxes that need to extend trust to an endpoint via other methods before the proposed system will mark that endpoint as trustworthy. Increasing this parameter increases reliance on alternative methods, as the number of initial trust-links that need to be established increases. It also slows down checks of trustworthiness, as the number of references that need to be checked increases. It is assumed that obtaining a significant number of malicious whiteboxes is difficult, as they should only be sold to GP's, and no GP should have more than one whitebox.

3.2.4 Operations

The whiteboxes make three operations available for their users. Other functions may be exposed for other whiteboxes to use in the administration of the system, but the user is limited to these three. We describe them each in turn here.

Check Trust

A whitebox can check the trust in an endpoint, figure 3.1 and 3.2 show a failed and a successful case of this respectively. The whitebox does so by obtaining a reference for S other whiteboxes which are noted as having a trust-link to a given endpoint from the discovery server. If the

whitebox cannot find a sufficient number there is no trust, and we must defer to UZI-backed endorsements to establish it. If the whitebox can find S peers with a trust-link to the endpoint, the whitebox asks those other whiteboxes for a trust report, each containing the number and nature(positive or negative) of trust-links the other whitebox believes an endpoint has. If the reports are favourable and mostly consistent, the whitebox trust the endpoint. This would allow for access to medical data, and an establish trust-link operation should follow.

Note that this paper does not propose a set amount of other whiteboxes necessary to establish trust. Rather, we leave S for configuration based on practical circumstances. Note that the minimum of necessary references establishes an upper bound for the amount of malicious whiteboxes necessary to fraudulently establish trust. Should an attacker control more whiteboxes than this number it would be possible to fraudulently establish a new endpoint as trusted, since benevolent whiteboxes will copy fraudulent trust-links backed by a sufficient number of others.

Establish Trust-Link

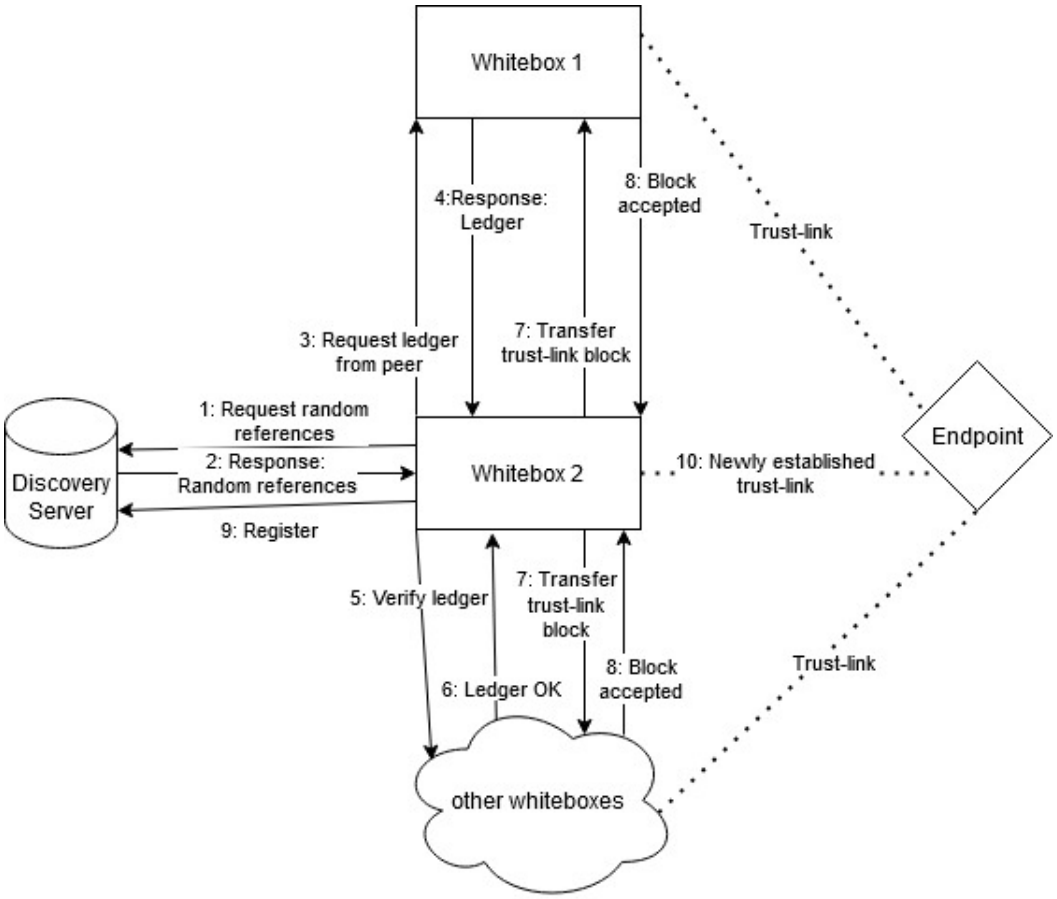

Figure 3.3: Whitebox 2 successfully establishes a new trust-link. Whitebox 1 is the randomly chosen peer where the ledger is downloaded from initially.

This operation, shown in figure 3.3, should be performed automatically after the whitebox has checked the trustworthiness of an endpoint and found it sufficient. The whitebox obtains a copy of the ledger from a random peer. It then compares hashes of the final blocks of the blockchain with several peers, to ensure the copy it obtained is correct. Should the whitebox performing the operation find that there is a conflict between differing versions, it will simply wait a few minutes and try again. If not, the whitebox can now publish a new block containing its trust-link establishment.

Since the ledger the whitebox has already obtained contains a list of all addresses of whiteboxes with a trust-link to the endpoint, the whitebox can now simply transmit its newly made block to all other whiteboxes in the ledger in order to register it's trust-link with them. Whiteboxes who receive this block verify that it is correct by seeing that the block is transmitted by the same whitebox whose trust-link the block captures. Any blocks with mismatching previous hashes or that describes the wrong whitebox will be rejected. Blocks with timestamps more than half an hour out of date will also be rejected.

The proposed method for ensuring consensus will sometimes lead to whiteboxes which differ from the common consensus, yet will not accept the correct block. For example, this can happen if the senders connection is slow, leading the operation to take over half an hour. If there are many misaligned whiteboxes, the operation is cancelled and the blocks are revoked. If there are only a few misaligned whiteboxes, a whitebox can send an "insist" message on a certain block.

Upon receiving a block that is insisted upon, a whitebox will check with at least half its peers and up to all of them, until it know for certain that half of them either accept or reject the block. It will then accept or reject the block accordingly itself. Therefore, this mechanism is not just useful for pushing a block that would not be accepted otherwise, but also to get a whitebox to reject an erroneous block it has stored. This is an expensive operation for large instances of the proposed system, as a large number of peers have to be contacted. It is only needed to correct rare errors however, which mitigates this problem.

Revoke Trust-link

There are two forms of this operation. The first version applies only to recent endorsements(an hour old or less). A whitebox can revoke these at will, invalidating the block and any blocks after that block. For any blocks discarded, their operations fail retroactively, and their associated whiteboxes will restart the operations once they receive the block revocation.

Otherwise, this operation is very similar to establishing a trust-link. The only meaningful difference is that the content of the block records a revocation rather than an establishment and that the address of the whitebox is removed from the ledger rather than added. The whitebox revoking the trust-link will also deregister itself from the discovery server.

Revoking a trust-link is also used to handle stopped or otherwise unreachable whiteboxes. For this purpose the publishing whitebox is different from the whitebox described. Receiving whiteboxes verify this block by trying and failing to reach the whitebox described.

3.2.5 Blockchain Conflict Resolution

For each new block, the whitebox publishing that block is responsible for making sure the block is accepted by all others. Block order is determined by timestamp. Should a whitebox's block publishing fail because of a block with an earlier timestamp than its new block, the whitebox copies and validates the problematic block, modifies its new block to go after it, and then restarts its publishing, now publishing both blocks. Should the whitebox encounter a block with a later timestamp, the whitebox receiving that block will remove the previous block and send a message to the owner of the block to inform it of the new block.

3.2.6 Dropping whiteboxes

It is possible for whiteboxes to spontaneously disappear from the system, without properly revoking their trust-links. Anytime a block is added to the blockchain, all such whiteboxes will inevitably be discovered. Then the discovering whitebox, after waiting and retrying the communication to ensure that the other whitebox is staying down, can publish a trust-link revocation block. Other whiteboxes can verify that this block is accurate by contacting the described whitebox themselves, and seeing that they get no response.

3.2.7 Public visibility

The proposed system can easily be extended to allow the ledger to be visible to the public. This can be done by establishing a public visibility server, which queries the discovery server for whiteboxes with a ledger for a given endpoint, then obtains a copy for itself as a whitebox might. It can then function as a webserver for a website displaying the contents of the ledgers obtained. While we do not implement this extension, its possible desirability does mean that data in the ledgers should be treated as publicly available.

3.3 Threat models and Adaptations

In this section we will be discussing relevant threat models. We will be covering malicious whiteboxes trying to establish additional trust or disrupt the system, denial-of-service attacks and inference attacks, where an attackers tries to use data available in the ledgers along with already known data to infer information about a patient. We will not be covering spoofing or message substitution attacks, as we assume them to be prevented by singing messages with the whitebox key described in section 3.2.1.

3.3.1 Malicious whiteboxes

We need to account for the possibility of one or more malicious whiteboxes. An attacker might attempt to use one or more whiteboxes they control to establish more trust than is due, or to perform a denial-of-service attack with access to privileged operations on the whiteboxes. Since data in the ledgers is assumed to be publicly visible (see section 3.2.7), we are not concerned with any additional data such an attacker might get by viewing the ledger.

For attempting to establish undue trust using malicious whiteboxes, the most likely method of attack would be to establish a new endpoint using the malicious whiteboxes as backers. This is essentially the same as establishing legitimate trust, the key difference being that the whiteboxes agree to establish trust-links to a fraudulent endpoint controlled by the attacker. The total number of trust-links cannot be used to guard against this, as benevolent whiteboxes will make new trust-links based of existing ones by malicious peers.

The key to preventing an attack of this nature is in choosing a high enough value for S. It should be high enough that an attacker shouldn't be able to gain control of S valid whiteboxes. If an attacker cannot gain control of S whiteboxes, they should be unable to establish invalid trust alone. See section 3.2.3 for more information on S.

The other type of attack enabled by having access to a malicious whitebox is a denial of service attack using priviledged functions. Specifically vulnerable to this is the insist mechanism used to ensure consensus. We can address this by extending the messages used to verify an insist with the address of the whitebox that originally insisted and a prediction that whitebox made. We can then bar the insist mechanism from whiteboxes which predicted incorrectly a certain number of times recently. Since insists are supposed to be rare, this should be possible. It should be noted that the insists mechanism is a naive way of ensuring consensus. In the section 5.2(Future Work) we discuss improvements that would also help mitigate this type of attack.

3.3.2 Denial-of-service attack

In a denial-of-service attack we assume that one or more devices that are part of the system become unavailable for a period of time. The system is resistant to whiteboxes becoming unavailable, assuming more than S whiteboxes stay up. The discovery server presents a weak spot when it comes to denial-of-service attacks. Should it go down, validating the trustworthiness of a new endpoint will no longer be possible. Known endpoints can still be validated, as the ledger contains the addresses of other whiteboxes with a trust-link.

To address this single point of failure, we suggest that at least one backup discovery server should be used at all times. The secondary server should copy the data of the primary one, and be available to handle the same requests. This increases the resilience to denial-of-service attacks, though multiple discovery servers could also be attacked simultaneously. This measure also helps prevent inaccessibility of the system due to downtime in the discovery server for other reasons, such as maintenance or power outages.

3.3.3 Inference attack

In an inference attack, an attacker makes use of accessible data to infer data other data that should be hidden. Given the possibility of making the ledger publicly visible (See section 3.2.7), it is a serious concern. For the proposed system, this can become possible if an attacker knows beforehand when a patient will visit a GP. If the whitebox belonging to the GP in question registers a new trust-link to an endpoint belonging to an organization at the same time, an attacker may infer that the patient will be visiting that organization. If that organization specializes in a specific type of medical problem, information on the patients health may also be inferred this way.

We can guard against this by decoupling the time of the trust-link formation from the time of GP interaction with the system. Rather than establishing a new trust-link immediately after verifying the trustworthiness of an endpoint, the whitebox should wait a randomly determined time between 1 and 24 hours. This obfuscates the connection between the registered trust-link and the interaction between the GP and the patient.

Trust check messages, potentially visible at the receiving whitebox, cannot be obfuscated in the same way and may therefore allow information to be inferred if captured by the malicious owner of a whitebox. We limit this possibility by randomizing the whiteboxes from which trust-reports are requested. This works well when there are many whiteboxes, but is not very effective when the number of whiteboxes is under or close to S.

CHAPTER 4

Prototype Analysis

In order to demonstrate the functionality of the proposed system, as well as to investigate scaling, we provide a prototype implementation. It is written in python using Flask and SQLite, each chosen to enable quick development. The prototype lacks several critical features, both pertaining to security and functionality. On the security side, it lacks any protection against impersonating a whitebox. On the functionality side, it lacks the ability to find consensus when there is a conflict, and support for removing inactive whiteboxes. It is therefore not recommended to apply this prototype in practice. It exists solely as a method for investigating the functionality and scaleability of the proposed system. We provide the source code of the prototype here: `https://github.com/MatthijsBartelink/healthcare_trust_ledger`

4.1 Test structure

We omit a thorough analysis of all operations available to the whiteboxes due to lack of time. Instead, we design tests to validate the primary design goal of fast trust checking and analyse the storage space used for each ledger. We omit analysis of the performance of the consensus algorithm, since conflict resolution is omitted from the prototype and can be substantially improved upon in future work.

4.2 Harddrive Space Usage

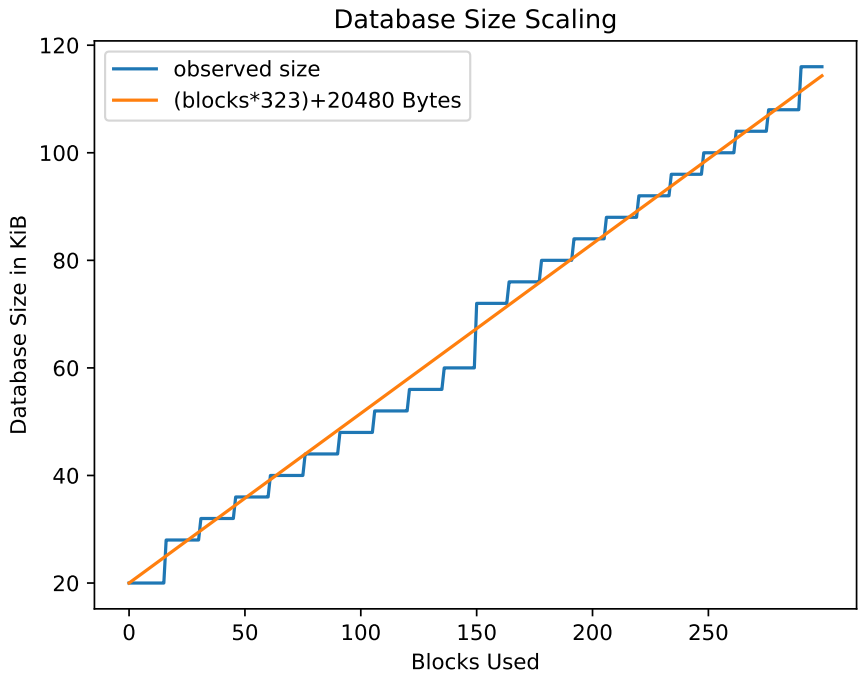

Figure 4.1: Observed database size compared to the number of blocks.

We measure the space taken up by the SQLite database file for the ledger for different counts of blocks. We show the measurements obtained this way in figure 4.2. We see that after a static initialization cost of 20 KiB, each block ends up taking up roughly 323 bytes. The jumps in observed size are a result of the way SQLite allocates pages, and is therefore implementation dependent.

This linear size scaling holds until at least 6000 blocks, which is the limit of the test performed. This is sufficient for the desired scale of roughly 5000 whiteboxes in a system, as revocations are expected to be rare. At 6000 blocks, a ledger takes up slightly less than 2 MiB. At the required capacity of 150 ledgers on a whitebox, roughly 300 MiB would be required.

4.3 Trustreport Generation Speed

We measure how the trust report generation speed scales with the number of blocks used. This gives an indication of the speed of checking the trust status of an endpoint. A whitebox needs to ask for a trust report from S different whiteboxes in order to check trust status. This test does not account for communication time between whiteboxes when checking trust status, as it measures local report generation time only.

The test consisted of repeatedly pushing a block, followed by measuring the time it takes to generate a trust report 100 times. We then record the median time. In terms of hardware, the server used in testing has a Intel(R) Xeon(R) E-2124 CPU @ 3.30GHz CPU and a HDD with a max transfer speed of 6 Gbps.

We find that the time taken scales linearly with the number of blocks in the ledger. This is likely a result of loading the blocks in the ledger in order to read their content. The linear

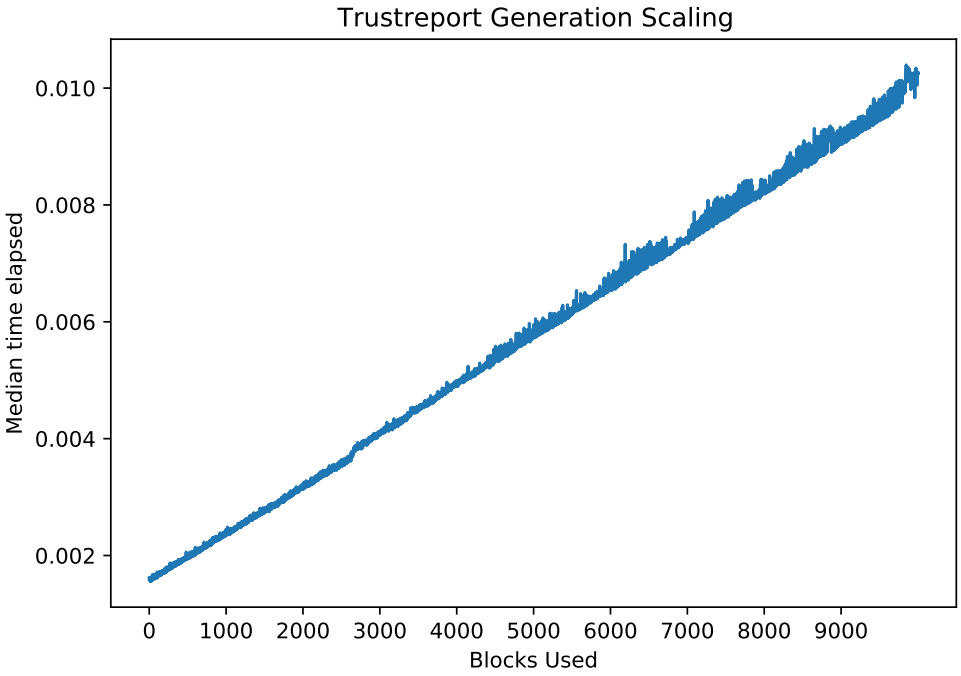

Figure 4.2: Trust report generation speed compared to the number of blocks.

scaling shows that the proposed system will scale sufficiently when it comes to trust check speed for the desired system size.

CHAPTER 5

Conclusion

5.1 Main results and findings

We propose an extension of the whitebox system, where whiteboxes gain the option of relying on previously established trust by other whiteboxes rather than always having to establish trust with an endpoint using endorsements. We accomplish this by keeping a decentralized proof-of-authority blockchain ledger which describes the current trust in a given endpoint, distributed among all whiteboxes which trust that endpoint.

We allow for the revocation of established trust-links by organizing the ledger as a sequence of addition and removal operations. This allows for trust links to be revoked by adding a new block to the ledger. We use a proof-of-authority consensus algorithm, where whiteboxes are the authorities on blocks describing their own trust-links. We tie the ledger to the identity of a specific endpoint by having the first block contain a hash of the address and the key associated with the endpoint.

Since allowing for the revocation of existing trust-links is insufficient to account for the loss of trust in an endpoint, we allow for negative trust-links to be registered. These negative trust-links function as a warning towards an endpoint considered untrustworthy by the whitebox. They are weighed more heavily compared to positive trust-links, allowing trust to be removed from an endpoint by fewer whiteboxes than it takes to establish.

We provide a prototype implementation using Python, Flask and SQLite. We then analyze this prototype in terms of scaling, disk-use and the speed of checking the trust status of an endpoint. Further performance analysis is omitted due to lack of time. We find that both disk-use and check speed are acceptable for the desired scale, and scale linearly with the number of blocks in a ledger.

The whiteboxes use a centralized discovery server to allow for the discovery of other whiteboxes with a trust-link to a previously unknown endpoint. This discovery server is a central point of failure for the system, particularly when it comes to denial of service attacks.

Another unfortunately centralized aspect of the proposed system is the reliance on Whiteboxsystems as the source of trust for the whiteboxes. This is currently necessitated by the proof of authority blockchain, which requires that whiteboxes are able to recognize each other as valid whiteboxes.

5.2 Future Work

Due to the limited timeframe available, there are several points in this research that can be considerably improved upon. The first priority for future work should be an implementation and thorough analysis of the consensus and conflict resolution algorithms. This would allow for realistic block publishing speed estimates. These algorithms also have significant room for improvement by applying random polling [1].

Another flaw future work may seek to improve upon is the discovery server. Rather than using a centralized server, it may be possible to have endpoints keep records of whiteboxes which trust them. This would allow these endpoints to function as the discovery server for their own ledger. This has security implications, and should be combined with a re-examination of the security of the proposed system.

A final improvement we suggest for future work is to decentralize trust in the whiteboxes. Rather than relying on centrally authorized keys to authenticate whiteboxes, it may be possible to set up a system of decentralized ledger storing endorsements to authenticate whiteboxes, similar to the ledger of trust-links proposed in the paper. This would address the reliance on Whitebox Systems as a source of trust.

Bibliography

[1] James Cruise and Ayalvadi Ganesh. "Probabilistic consensus via polling and majority rules". In: *Queueing Systems* 78.2 (2014), pp. 99–120.

[2] Mohamed Tahar Hammi et al. "Bubbles of Trust: A decentralized blockchain-based authentication system for IoT". In: *Computers & Security* 78 (2018), pp. 126–142. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2018.06.004. URL: http://www.sciencedirect.com/science/article/pii/S0167404818300890.

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

[4] Guido van't Noordende and Bas Kloosterman. *identiteit en authenticatie: onze zorg*. Whitebox Systems, Een rapport in opdracht van Stichting beter met elkaar, 2021.

[5] Vishal Patel. "A framework for secure and decentralized sharing of medical imaging data via blockchain consensus". In: *Health Informatics Journal* 25.4 (2019). PMID: 29692204, pp. 1398–1411. DOI: 10.1177/1460458218769699. eprint: https://doi.org/10.1177/1460458218769699. URL: https://doi.org/10.1177/1460458218769699.