Zero Knowledge Validation System With Multiple Decentralized Data Providers

Mathias Maier

MASTERARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im Juni 2019

Declaration

I hereby declare and confirm that this thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere.

Hagenberg, June 24, 2019

Mathias Maier

Contents

Abstract

Zero Knowledge Proofs in combination with Blockchain enable the possibility for new applications in which a user does not have to give away sensitive data for data processing but still let external instances validate results processed with the user's data. With this users can be in complete control of their data. Data breaches or similar threats can be minimized with this due to keeping data on each users device instead of storing it centralized in a single database. The first part of this master thesis is about the design and implementation of an architecture focused on a Perfect Zero Knowledge system which meets the requirements of Zero Knowledge Proofs. The prototype is simulated within a perfect environment in which every instance such as data providers or users implement and work with the prototype and offered interfaces.

Due to the General Data Protection Regulation (GDPR) companies have to implement mechanisms such as Privacy by Design to be compliant with the GDPR. The second part of the thesis is about the evaluation of the prototype and architecture against the 7 Principles of Privacy by Design to show if such a system can comply with it. An elaboration will show why and how principles are fulfilled or which parts of such an architecture may need to be changed to be compliant. This evaluation is done for the prototype within the perfect environment. As a second evaluation a scenario when having the architecture in a live environment is also evaluated to display if principles can be fulfilled within the test but not in the live environment.

Kurzfassung

Zero Knowledge Proofs in Kombination mit Blockchains ermöglichen neue Anwendungen, in denen ein Anwender keine sensiblen Daten für die Datenverarbeitung preisgeben muss, aber dennoch externe Instanzen die mit den Daten des Benutzers verarbeitete Ergebnisse validieren lässt. Damit haben die Anwender die volle Kontrolle über ihre Daten. Datenschutzverletzungen oder ähnliche Bedrohungen können so minimiert werden, da die Daten auf jedem Endgerät gespeichert werden, anstatt sie zentral in einer einzigen Datenbank zu speichern. Der erste Teil dieser Masterarbeit beschäftigt sich mit dem Design und der Implementierung einer Architektur, die sich auf ein Perfect Zero Knowledge System konzentriert, welche die Anforderungen von Zero Knowledge Proofs erfüllt. Der Prototyp wird in einer perfekten Umgebung simuliert, in der alle Instanzen wie Datenanbieter oder Anwender den Prototyp implementieren und damit arbeiten.

Aufgrund der Datenschutz-Grundverordnung (DSGVO) müssen Unternehmen Mechanismen wie Privacy by Design implementieren, um mit der DSGVO konform zu sein. Der zweite Teil der Arbeit beschäftigt sich mit der Evaluierung des Prototyps und der Architektur anhand der 7 Prinzipien von Privacy by Design, um zu zeigen, ob ein solches System die Voraussetzungen erfüllt. Eine Ausarbeitung zeigt, warum und wie Prinzipien erfüllt werden oder welche Teile einer solchen Architektur geändert werden müssen, um konform zu sein. Diese Evaluierung erfolgt für den Prototyp in einer perfekten Umgebung. Als zweite Evaluierung wird ein Szenario mit der Architektur in einer Live-Umgebung ausgewertet, um zu zeigen, ob die Prinzipien innerhalb der Test-Umgebung, aber nicht in der Live-Umgebung erfüllt werden können.

Chapter 1

Introduction

The following chapters will give some background information as well as a general overview of why the topic is relevant to be researched. Further the goal of the thesis is described itself along with how it will be achieved.

1.1 Motivation

The decentralized architecture of Blockchains together with its increasing maturity enable the possibility for new so called decentralized applications or dapps. Those applications are not running in centralized instances but in a decentralized environment like on a Blockchain. The trust is, contrary to centralized applications, validated and distributed to a lot of devices instead of having a single point of trust. This means that the majority of the devices need to approve certain actions to be validated.

Currently some companies exist which are trusted by institutions. For example tax agencies which calculate realized gains of cryptocurrency trades are trusted by the government to provide the correct (not modified) tax result. If a user wants to use this offer one has to give away data which is needed for example the calculation. Zero Knowledge Validation, in combination with the matured Blockchains, enable the possibility for new applications in which a user does not have to give away sensitive data but still let external instances validate results processed with the user's data. Although the research and concept of Zero Knowledge as an Interactive Proof System has already been introduced in 1985 by Goldwasser, Micali and Rackoff it has now the potential to increase online privacy together with security [13]. Data breaches or similar threats could be minimized with this due to keeping data on each users device. Because of the new EU General Data Protection Regulation (GDPR)[1] introduced in 2018, companies also have to implement privacy by design as a default mechanism. This could go hand in hand with the decentralized applications including validation among zero knowledge proofs.

1.2 Problem Statement

The amount of data breaches, leaks of company data, furthermore as a consequence also the exposure of registered and managed users has been increasing a lot in the last

[1] https://gdpr-info.eu/art-25-gdpr/

years. According to Gemalto, the world leader in digital security, 945 data breaches have been executed in the first half of 2018 which led to 4.5 billion data records being compromised [22]. This is an increase of 133 percent when comparing it to the same period of 2017. The avast report about the biggest data breaches in 2018 support this statistic by showing that the data of 2.9 billion people have been leaked containing names, email addresses, credit card information etc. [23].

People are sharing a lot of information without even knowing what happens with the data given away. One recent example of this is the Cambridge Analytica data leak which affected 87 million users [25]. The GDPR should counteract such situations by engaging companies to explain people in a structured way how user related data is stored and processed. Infringements result in high fees for the companies. This enables more transparency. Unfortunately, due to the architecture of current systems, a lot of data is stored in a single storage unit such as databases or managed online cloud services. This leads to a single point of failure when willingly or unwillingly exposing login information or simply being hacked. Decentralized Zero Knowledge Systems which comply with the GDPR and privacy by design may solve this problem.

1.3 Research Question

Based on the results of this evaluation the following research question will be answered: What aspects of privacy by design are complying with Zero Knowledge Systems?

1.4 Aim of the Work

The goal of the work is the utilization of a Blockchain to implement a fully working Zero Knowledge validation system prototype which serves as a basis to answer the research question. It should also assist for a comparison to existing approaches in which differences and problems may be pointed out. The system should be able to work with multiple data providing instances (data provider), a validating instance (validator) and proving instance (prover). In the system, the prover has to prove the validator that the result of a calculation with the data provided by the data provider is valid without revealing which data was used. The Blockchain is used as a decentralized storage which is publicly accessible by every instance.

The implementation of the prototype is followed by the validation against the principles of privacy by design [6]. It should be pointed out which aspects can be fulfilled but also which cannot be fulfilled including an analysis why they cannot be satisfied. For the aspects which are not complying with the prototype a theoretical approach to solve the issue is suggested if possible.

1.5 Structure

The thesis itself is structured in three major parts. The first part gives information concerning the general problem as well as background information, continued by the fundamental technologies to get basic knowledge as well as the state of the art to show existing approaches in similar fields.

The second part is about the technical analysis of the problem, its theoretical solution approach and furthermore the problems along with the practical implementation. The thesis is finalized in the third part which discusses the analysis of the implementation in contrast to the existing approaches as well as the verification of privacy by design to answer the research question. Potential follow up questions should show open issues for further research.

1.6 Methodology

The goal of the thesis in terms of methodology for the prototype is to implement a fully working Zero Knowledge Validation system with as much open source software as possible to speed up the development process. This is important for example the implementation of the blockchain. However, the validating and proving processes are customized to fit the situation.

Because the principles of privacy by design are already existing, the prototype is validated against each single principle and analysed why it is or is not fulfilled. A negative validation may lead to a comparison with existing solutions which are analysed in an earlier step. Not only the validation of each principle but also internal conflicts of the principles with the prototype are pointed out.

Chapter 2

Fundamentals

To get a basic understanding about the topics thematized in this thesis the following chapter gives an overview and simple explanation about the technology used.

2.1 Blockchain

Blockchains are best described as Crosby summarizes it in [7]:

> *A blockchain is essentially a distributed database of records or public ledger of all transactions or digital events that have been executed and shared among participating parties. Each transaction in the public ledger is verified by consensus of a majority of the participants in the system. And, once entered, information can never be erased. The blockchain contains a certain and verifiable record of every single transaction ever made. To use a basic analogy, it is easy to steal a cookie from a cookie jar, kept in a secluded place than stealing the cookie from a cookie jar kept in a market place, being observed by thousands of people.*

When speaking about the Blockchain, the longest sequence of blocks is meant which is represented as the black blocks in Figure 2.1. The genesis block which is displayed in green defines the first block of an chain. Other than all other blocks this one is not calculated by the network but defined in the source code. Not accepted blocks or side chains are represented as purple blocks. Those side chains are defined as orphans which are not accepted from the blockchain network due to a time lag in the acceptance process of the block. So orphan blocks are valid and verified blocks but rejected by the network.

In terms of accessibility Blockchains can be differentiated in three different types: Public Blockchains, Private Blockchains and Consortium Blockchains. Public Blockchains are best known for being used in the Bitcoin Cryptocurrency application[1]. This type is publicly available for everyone which leads to high transparency. Private Blockchains which are run by a single organisation are permissioned and only available if access is granted. This type is used when for example members of an organisation are geographically distributed and cannot trust each other completely. The Blockchain is closed and

[1] https://bitcoin.org/bitcoin.pdf

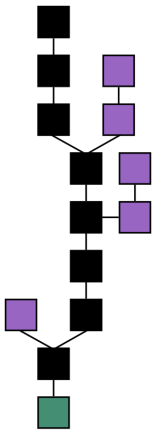


Figure 2.1: Simplified representation of a Blockchain.

only transparent in the organisational unit. Consortium Blockchains are similar to Private Blockchains. This type is also permissioned but instead of being controlled by one single organisation the Blockchain is controlled by a consortium of companies [21].

Due to the change of requirements and development of more complex applications the Blockchain itself evolved as well. The first version of the Blockchain was all about currencies and famous because of the use in the Bitcoin application. However newer Blockchains in the second generation such as Ethereum allow users to develop applications on top of the Blockchain. As described by Swan in [16] this means that one Blockchain infrastructure can serve multiple applications and is not limited to simple cash transactions. For example the economic market and financial applications such as stocks or properties can be handled within so called smart contracts which are running on the Ethereum Blockchain.

2.2 Zero Knowledge Proof

The system can be considered as a perfect zero knowledge system. The Simulator S can reproduce the proof perfectly

The concept of Zero Knowledge Proofs which was introduced by Goldwasser, Micali and Rackoff in 1985 is an extension of interactive proof systems. In an interactive proof system the prover and verifier interact by exchanging messages in an given order to check if a given message can convince the proof system to be a valid message or not. In theory the prover has unlimited computing power and cannot be trusted. The verifier has limited power with which the message has to be validated. While Feige describes Zero Knowledge Proofs as “an elegant technique to limit the amount of information transferred from a prover A to a verifier B in a cryptographic protocol” [10] Goldreich extends this with “zero knowledge proofs are proofs that yield nothing beyond the validity of the assertion” [12]. This of course means that due to only sending validity hashes and not the actual data the amount sent is minimal.

Generally speaking it can be said that the zero knowledge protocol or zero knowledge

proof is an interactive proof system (P, V) in which one party (the prover P) can prove to another party (the verifier V) that something is $true$, without revealing any information apart from the fact that this specific statement is $true$. "In other words; zero knowledge proofs let you validate the truth of something without revealing how you know that truth or sharing the content of this truth with the verifier. This principle is based on an algorithm that takes some data as input and returns either $true$ or $false$" [29]. So it should always be shown that the input x is part of the formal Language L. The Zero Knowledge Proof Protocol has three important properties which have to be fulfilled to provide a valid efficient system: completeness, soundness and zero knowledgeness. Completeness and soundness are properties of a more general interactive proof system. The zero knowledgeness makes the interactive proof system to a Zero Knowledge Proof System. Efficient in this term means that the verifier should be able to run in time polynomial in the length of the assertion[2].

Completeness: The proof system is complete if all true statements can be validated with the system. If all statements are $true$, the honest verifier will be convinced with a high probability of this fact by an honest prover. If the input x is part of the formal language L ($x \in L$) the verifier should accept the statement.

Soundness: When the interactive proof system can never derive a $false$ statement using the system it means that the system is sound. If the statement is $false$ no cheating prover can convince the honest verifier that the statement is $true$. In practical use this property might not be 100% fulfilled because there is a minimal probability that a guess can be correct. So if the input x is no part of the formal language L ($x \notin L$) the statement should be refused but a small error probability is allowed.

Zero Knowledgeness: Having completeness and soundness fulfilled, the Zero Knowledge Proof also requires that no verifier learns anything other than the fact if the given statement is $true$ or $false$. Underlying information is only available for the prover and strictly hidden from the verifier.

Due to the small probability of the soundness error one can say that zero knowledge proofs are more probabilistic proofs rather than deterministic proofs. That small probability could let a cheating prover convince a verifier that a $false$ statement is $true$. There are three variants of zero knowledge Systems which have a different probability of the soundness error and can be used in different scenarios.

2.2.1 Statistical Zero Knowledge

Statistical zero knowledge(SZK) as stated in [17] requires that the distributions are statistical close. SZK are Zero Knowledge Proofs in which the term, that the verifier learns nothing is interpreted in a strong statistical sense. Meaning that there can be a difference which is a negligible function.

[2] "An algorithm is said to be solvable in polynomial time if the number of steps required to complete the algorithm for a given input is $O(n^k)$ for some nonnegative integer k, where n is the complexity of the input." as described in [31]

When using SZK the verifier challenges the prover. The prover then tries to solve the challenge. This process is repeated over and over again until the verifier is convinced. The goal is to keep the soundness error below for example 1 percent. This means that the given challenge needs to be accepted 99 out of 100 times.

2.2.2 Perfect Zero Knowledge

For a language to have a perfect zero knowledge proof, a proof of membership to the language needs to be exhibited so that the proof can be simulated perfectly without knowing the witness. Even an unbounded adversary should not be able to see any difference between the real proof and the simulated proof.

Perfect zero knowledge works by carefully exploiting the structure of the problem at hand. For example, the best way to prove that a language L, for which no efficient algorithm is known, involves exhibiting a perfect zero knowledge proof for it [11]. The definition states that for every Verifier V there exists a Simulator S which can reproduce the conversation between V and P perfectly.

2.2.3 Computational Zero Knowledge

Goldwasser, Micali and Rackoff mention in [13] that "Computational zero knowledge in comparison to statistical or perfect zero knowledge is the most general of the zero knowledge systems. It only requires that the simulated proof cannot be distinguished from the real proof by a computationally bounded adversary" . This statement is supported and simplified in [28] by saying that a "common reference string which is shared between the prover and the verifier [...] is enough to achieve computational zero knowledge without requiring interaction".

2.3 Trustless Validation

Blockchains are sometimes referred to be trustless but they do not actually eliminate trust. It is more of a verifiable computation on a decentralized system in which the trust is distributed. The amount of trust required from any single actor in the system is minimized but distributed among a lot of different actors in the system (distributed trust) as visualized in Figure 2.2. If for example Bitcoin is transferred from Person A to Person B the transaction is not validated by a single actor but a defined amount of actors or consensus [24]. If the majority agrees on the validity the transaction is accepted, otherwise it is rejected.

Zero Knowledge proofs can be realised as, for example, smart contracts on the Ethereum blockchain. Smart contracts are applications which are running on a blockchain. The functions, which are executed on the blockchain for this project, are also distributed on the blockchain. There is no centralized server and the source code is publicly visible if the blockchain is public. This enables a high transparency and allows parties to interact with each other even though they do not trust each other. For example the smart contract for the Blockpit TAX Token™[3], can be analysed on Etherscan[4], which is a

[3] https://etherscan.io/token/0xe9990eda9e478ded1f8318d7002ed08d5073e71d#readContract
[4] https://etherscan.io

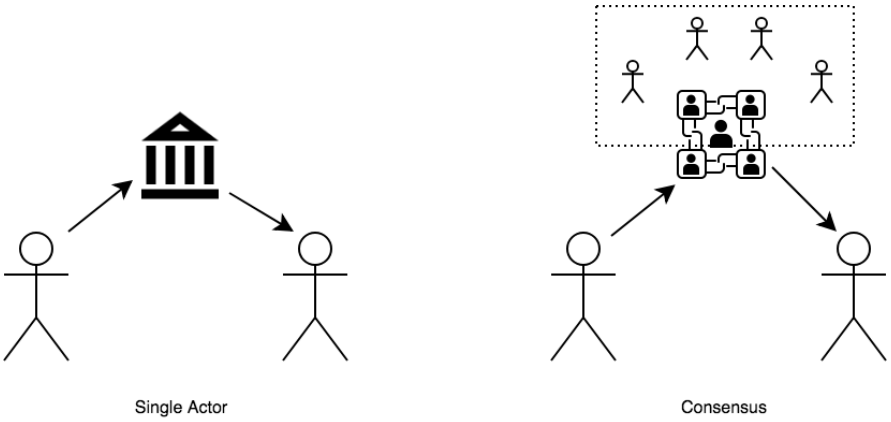


Figure 2.2: Simplified comparison of validation mechanism.

platform to scan the Ethereum blockchain and gather information from transactions and applications. The applications cannot be manipulated after the first deployment due to the fact that smart contracts are immutable. This ensures users that the source code is not changed or manipulated but leads to more expensive update processes.

2.4 Privacy by Design

The term "privacy by design" was initially formalized by Ann Cavoukian, the Information and Privacy Commissioner for Ontario, in 1995 and released as a framework in 2009 [6]. Together, and as a part of the framework, a set of guiding principles in the design of computer software has been published. Privacy by design changes the process when engineering for example software applications. Other than just writing the application, human values should be taken in account. The applications should be human centric in a privacy meaning during the whole engineering process. Every protocol and process with which humans interact must embed privacy. With the release of the European GDPR regulation in May 2018, the European Data Protection Supervisor (EDPS) endorsed the principles of privacy by design and calls on the European Parliament to "support privacy when adapting or creating legal frameworks which influence the design of technology, by increasing incentives and substantiating obligations, including appropriate liability rules, to integrate privacy by design in products and services" [9]. However the principles also have been criticized by Rubinstein for being vague and difficult to apply to processes [14]. Privacy by design also only defines how processes should be engineered to store and collect data. It only regulates that the data is collected and stored safely but not in an ethical sense if the data should even be used. The seven principles of privacy by design by Ann Cavoukian as described in [6] are summarized in the following paragraphs.

Proactive not Reactive; Preventative not Remedial: One should prevent privacy invasive events proactive before they even happen. Privacy by design does not wait for attacks

and problems to happen. It also does not offer remedies when resolving privacy invasions after they occured. For this a privacy commitment should be available and shared to the users, communities and stakeholders. This includes continuous improvement to correct any negative impacts once they are known.

Privacy as the Default: Privacy by design seeks to ensure that all personal data is automatically protected in any system or practise. The privacy of an individual is given by the person to protect the privacy. This principle can be divided to the following Fair Information Practices (FIPs).

- Purpose specification: The purpose why which personal information is collected, used, retained and disclosed should be communicated to the person at the time or before the time the data is collected.
- Collection Limitation: Only data which is necessary for a specified purpose should be collected.
- Data Minimization: Only a minimum of personal data should be collected. The linkability of personal information should be minimized.
- Use, Retention and Disclosure Limitation: The use, retention and disclosure of personal information should be limited to the relevant purposes for the individual.

Privacy Embedded into Design: Privacy by Design becomes an essential component of the core functionality being delivered. It is strictly embedded into business practises and technological systems. Not only IT systems need to implement privacy in the design process but also stakeholders and interests should be consulted. This also means that some (existing) choices might be unacceptable and have to be overthought.

Full Functionality – Positive-Sum, not Zero-Sum: To have a "win-win" manner when adapting Privacy by Design no unnecessary trade-offs should be made. No functionality should be discarded. When embedding privacy it should be done in a way in which full functionality is not impaired. Neither the company nor the individual should have an unnecessary trade-off.

End-to-End Security – Lifecycle Protection: Every element of information which is collected has to be transferred securely through the entire lifecycle in which the data is involved. All data needs to be securely retained and securely destroyed after the end of the lifecycle. Gaps in either protection or accountability are not acceptable.

Visibility and Transparency: The goal is to assure all stakeholders that business practises or technologies used are operated accordingly to the stated promises. The component parts and operations are transparently communicated to all involved entities. For auditing purposes, special emphasis is discussed in the following FIPs.

- Accountability: It is necessary to document and communicate privacy-related policies and procedures. This task needs to be assigned to a specified individual. When transferring private data to third parties the data should be secured through contractual means.

- Openness: Policies and practises relating to the management of personal data should be communicated in an open and transparent way.
- Compliance: Steps to monitor, evaluate and verify compliance should be taken as well as establishing complaint and redress mechanisms.

Respect for User Privacy:

> *Above all, Privacy by Design requires architects and operators to keep the interests of the individual uppermost by offering such measures as strong privacy defaults, appropriate notice, and empowering user-friendly options. The goal is to ensure user-centred privacy in an increasingly connected world. Keep it user-centric.* [6]

Empowering individuals to play an active role in the management of personal data might be one of the most effective ways to prevent misuses and abuse. For this, individuals should have access to their data and be able to challenge the accuracy and completeness at any time. Organizations, on the other hand, must adopt processes for complaints and redresses.

Chapter 3

State of the Art

Few applications using Zero Knowledge Proofs with a blockchain proof that such systems are possible by now. Although applications exist in live environments none of them works with multiple sources and validation instances.

3.1 Analysis

The increased interest of blockchain and cryptocurrencies in 2016 lead to more research in the area of zero knowledge proofs within blockchain applications. Not only because it is a very useful technology in this area but also to improve existing cryptocurrencies in terms of anonymity. Bitcoin for example it is often misled to be transparent and anonymous. In reality it can be easy to connect transaction to an individual as described by Robinson in [27]:

> *Any bitcoin transaction with a party that knows your identity leaks information that can be used to identify your activity, past and future, on the block chain. For example, if you transfer bitcoins to an online retailer, an exchange, or many of the other services that take customer identity information, you allow them to link that identity to your block chain pseudonym, potentially revealing the other transactions that you are party to.*

In addition to this statement it can be said that the Bitcoin transactions are not confidential because the amount sent is publicly visible. Graphsense[1], a cross-ledger cryptocurrency analysis platform developed by the Austrian Institute of Technology, for example allows a deep analysis of Bitcoin wallets and it's transactions. It is also possible to cluster wallets and determine which wallets belong to the same user due to their transaction history as already stated earlier.

[1] https://graphsense.info/

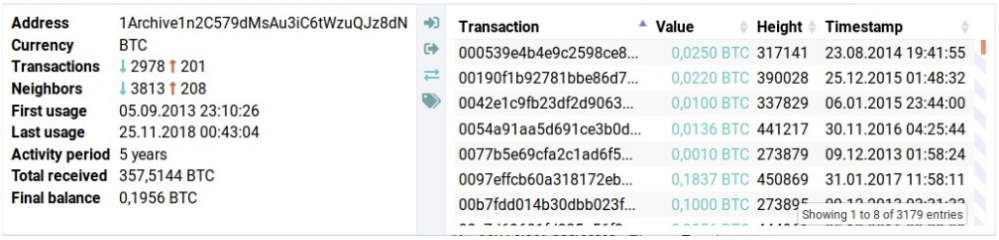


Figure 3.1: Graphsense allows deep analysis and investigation of Blockchain wallets.

Because of those reasons some newer cryptocurrencies attack the weaknesses and implement Zero Knowledge validation processes to validate transactions in which the user decides which parts are publicly visible. The detail about privacy protecting digital currencies is discussed later in Section 3.1.1.

One way to implement such a validation process can be with done with the implementation of zk-SNARKS. The term zk-SNARK or zero knowledge succinct non-interactive arguments of knowledge was introduced in 2012 by Bitansky, Canetti Ciesa and Tromer in [2]. SNARKS can be explained as the ZCash Foundation describes it in [30]:

> *The acronym zk-SNARK stands for "Zero-Knowledge Succinct Non-Interactive Argument of Knowledge," and refers to a proof construction where one can prove possession of certain information, e.g. a secret key, without revealing that information, and without any interaction between the prover and verifier.*

Because SNARKs have a few drawbacks some other technologies[2] have been evolved out of it. zk-STARK (zero-knowledge succinct transparent argument of knowledge) for example have been introduced in 2018 [8] and are sometimes referred to as the improved version of zk-SNARKs which fixes some problems. One major problem SNARKs have is the initial trusted setup. "A trusted setup means you need to trust someone to generate some initial parameters and then destroy those parameters" [32]. This also means that if an attacker manipulates the initial parameter creation ceremony it could be used to generate fake proofs. "zk-SNARK proofs are dependent on an initial trusted setup between a prover and verifier, meaning that a set of public parameters is required to construct zero-knowledge proofs and, thus, private transactions. These parameters are almost like the rules of the game, they are encoded into the protocol and are one of the necessary factors in proving a transaction was valid. However, this creates a potential centralization issue because the parameters are often formulated by a very small group" as described in [18].

This might be a problem for completely decentralized applications which need no trusted entity such as cryptocurrencies but might be no problem for applications which need a trusted entity anyways. This initial ceremony is not needed for STARKS. "As an alternative version of zk-SNARK proofs, zk-STARKs are, generally, considered a more efficient variant of the technology — potentially faster and cheaper depending on the implementation. But more importantly, zk-STARKs do not require an initial

[2] https://zkp.science/

Table 3.1: A comparison of SNARKs, STARKs and Bulletproofs.

	SNARKs	*STARKs*	*Bulletproofs*
Algorighmic complexity: prover	$O(N * log(N))$	$O(N * poly - log(N))$	$O(N * log(N))$
Algorighmic complexity: verifier	$O(1)$	$O(poly - log(N))$	$O(N)$
Size estimate for 1 transaction	200 bytes	45kb	1.5kb
Size estimate for 1000 transactions	200 bytes	135kb	2.5kb
Trusted setup required	yes	no	no
Post-quantum secure	no	yes	no

trusted setup" [18]. There are also problems theorized which could lead to problems in the future with for example quantum computers. According to Eli Ben-Sasson et al. in [8] the STARKs are also post-quantum computer resistant by stating "the existence of collision-resistant hash functions for interactive solutions, and common access to a random function for noninteractive ones — are not known to be susceptible to attacks by large-scale quantum computers; we call such solutions post-quantum secure".

Another technology which has been developed are Bulletproofs which were introduced in 2017 by the Stanford University [5]. Bulletproofs compared to STARKs are slower but have smaller proof size. It is also no trusted setup for the parameter generation required but since the algorithm is not post-quantum secure it has a weakness against STARKs. When comparing Bulletproofs to SNARKs it can be said that the proof is bigger and the validation slower but it has the advantage that no trusted setup is needed. Bulletproofs are created to create confidential transactions in which the amount transferred is hidden. One application which utilizes this is for example the cryptocurrency Monero[3].

The github user gluk64 did an comparison of SNARKs, STARKs and Bulletproofs as shown in the Table 3.1[4]. The data is mostly supported by the Defcon presentation [19] of Elena Nadilinski but she adds that the proof size of STARKs could be up to 200KB. This means the big proof could actually 4 times as big as displayed by gluk64. This of course might be depending on the implementation.

3.1.1 Privacy-protecting, Digital Currency

The cryptocurrency Z-Cash was introduced in 2016 and is an improved Version of the Bitcoin protocol in terms of privacy and anonymity[5]. The Z-Cash application runs on its own public blockchain and can do everything Bitcoin can. However it allows users to have different types of transactions which are realized as zero knowledge SNARK circuits as visualized in Figure 3.2.

[3] https://www.getmonero.org/

[4] https://github.com/gluk64/awesome-zero-knowledge-proofs

[5] Z-Cash builds upon the source code of Bitcoin.

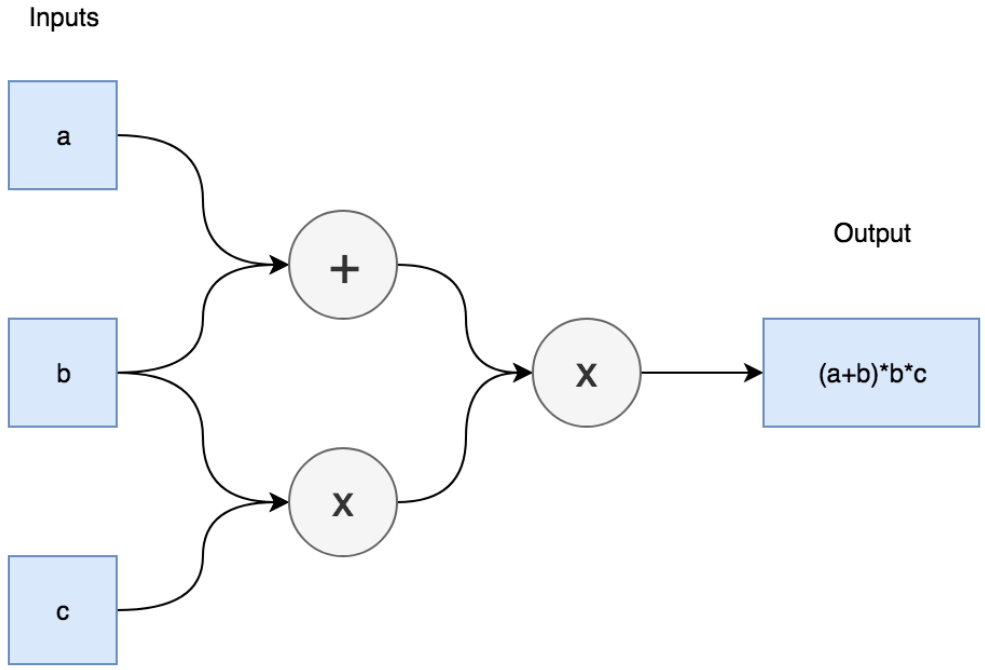


Figure 3.2: Example of what an arithmetic circuit looks like for computing the expression $(a + b) * (b * c)$.

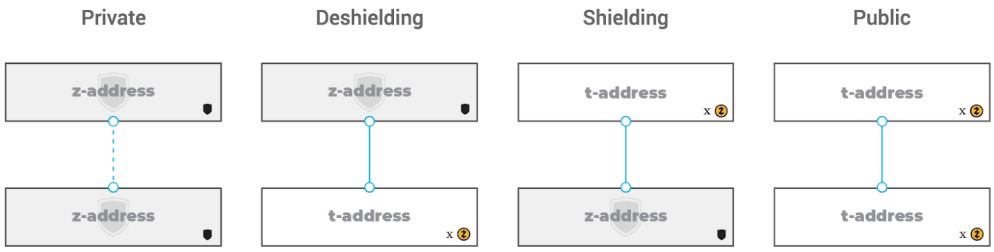


Figure 3.3: Different types of Z-Cash transactions which allows the user to define the privacy [30].

SNARKs, as explained in Section 3.1, are the backbone of Z-Cash. The cryptocurrency provides users the possibility to generate private or public wallet addresses. Private addresses start with z and are z-addresses whereas public addresses start with t and are t-addresses. With the different addresses Z-Cash supports different types of transactions as visualized in Figure 3.3 such as "shielded (private) transactions where sender, receiver and amount are not revealed; and yet, an outside observer can still distinguish between a valid and non-valid transaction" [3]. In difference to a public to public transaction details of the transaction are encrypted on the blockchain and only visible for participants of the transaction. This is possible because of Z-Cash's implementation of

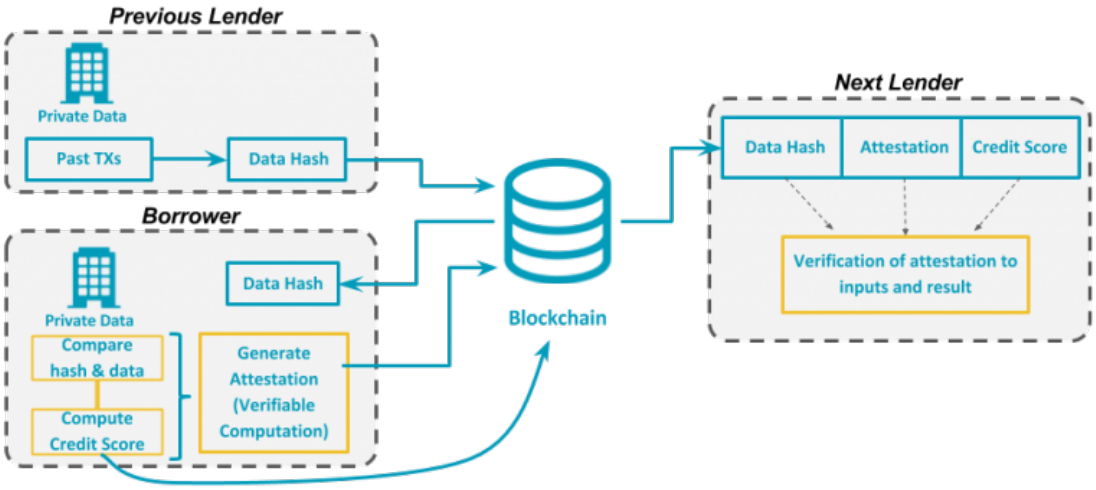


Figure 3.4: The implementation of QED-it's validation system visualized [20].

Zero-Knowledge SNARKs, which are scripts running on their blockchain.

However because of the use of SNARKs and its need for a trusted setup Z-Cash might have some weaknesses as well. "A potential weakness of Zcash, is that if anybody obtained the trapdoor information corresponding to the Common Reference String (CRS) used for constructing and verifying the SNARKs, they could forge unlimited amounts of the currency, potentially without anyone detecting they are doing so" [3].

3.1.2 Trustless Computing on Private Data

QED-it, a pioneer in Zero Knowledge blockchain applications, introduced a use case for money lending companies as visualized in Figure 3.4. While a borrower has to provide a credit score, the lender can validate if the credit score is computed from the borrower's past transactions and correct with no underlying data revealed [20]. This is possible with QED-it's implementation and interpretation of a blockchain based Zero Knowledge System.

Other than the Z-Cash application as described in Section 3.1.1 a hashed version of the last calculation is stored in the blockchain and visible for everyone. This data hash consist of the previous data hash, the attestation and the credit score. With the information received from the borrower the next lender can validate if the borrower said the truth or not with the hashes stored on the blockchain by generating the same hash. This can be interpreted as a perfect zero knowledge simulator as mentioned in Section 2.2.2.

One challenge which is mentioned is the generation of the verifiable computation or attestation on the users device. This is theoretically approached via cryptographic commitments in the blockchain to ground the underlying data or use verifiable computation schemes to prove the computation is correct without revealing the data. zk-SNARKs could be one way to solve this. Another possible way would be the use of for example obfuscated applications or let the application itself run in an enclave such as the SGX

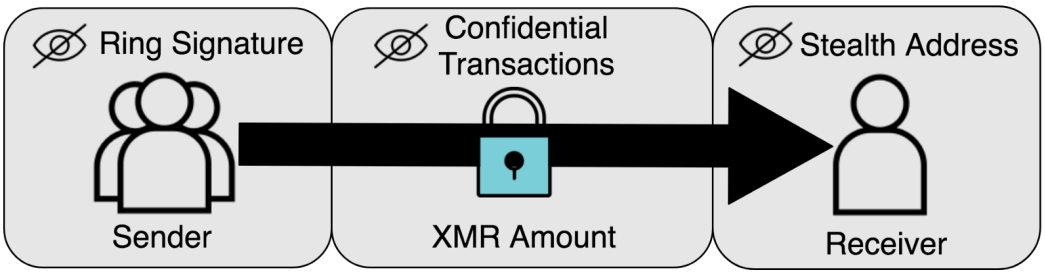


Figure 3.5: The three key parts of a Monero transaction [26].

Enclave. When using the SGX Enclave[6] this would require that every user has an Intel Processor but allows to run the application in an encapsulated area. It should not be possible to manipulate for example a calucation which is happening inside the enclave. However using the enclave or an obfuscated application also have drawbacks. Obfuscated applications could theoretically be reverse engineered and attacks on enclaves have already been demonstrated such as the SGX Cache attack [4].

3.1.3 Bulletproof Transactions

The cryptocurrency Monero[7] utilizes thee key pieces of technologies as displayed in Figure 3.5 to perform transactions. Similar to the cryptocurrency Z-Cash, as explained in Chapter 3.1.1, this currency allows private transactions which are not traceable. With the first piece of a transaction, the Ring Signature, it is possible to generate a signature which is needed for Confidential Transactions when sending the cryptocurrency XMR to Stealth Addresses. Stealth Addresses are similar to private Addresses of z-Cash.

The Ring signature is the major part to validate transactions without revealing the identities and amounts of the sender and receiver. The true sender of a transaction combines, instead of having just the data of the own transaction, the signature with multiple other signatures from past transactions. This allows to create a unified digital signature as displayed in Figure 3.6 which can only be reproduced by the real signer. Rather than a single identity, this unified digital signature represents a group.

Monero began to implement Bulletproofs. With this technology it is possible to address a major problems which occurs when generating Ring Signatures: the size of the proofs. The non interactive Zero Knowledge Proof Bulletproof does, on the one hand, not require a trusted setup to generate the parameters and have a much smaller size relative to the proof systems. However the verification of the newly generated proof is more time consuming. In fact, Bulletproofs can be seen as an approach to vertical scalability as they can greatly decrease the size of a cryptographic proof from over 10kB to less than 1kB [26].

[6] https://software.intel.com/en-us/sgx
[7] https://www.getmonero.org/

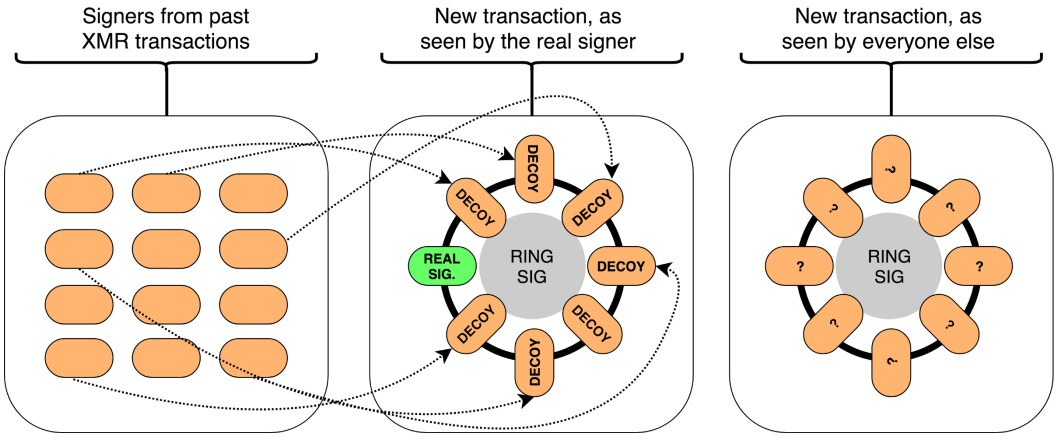


Figure 3.6: Moneros ring signature scheme [26].

3.2 Comparison of existing approaches

Z-Cash, as described in Section 3.1.1, has a quite different field of use than the borrower lender system, as described in Section 3.1.2, but a similar field as the Bulletproof transactions of Monero of Section 3.1.3. However when analysing the three they have pros and cons when using their own blockchain. While z-Cash and Monero are using their own blockchain they can adapt and optimize it to fit their need in a near perfect way. However a major task will be maintaining the distribution of the blockchain to get and keep the trust that it is really distributed decentralized. When compared to Z-Cash, the application of QED-it is build on an existing infrastructure. With this the overhead is lower and the development can concentrate on the core application. However this could lead to problems because nothing can be done when the blockchain does not exist anymore or the blockchain looses trust because of for example an attack on the network. With a backup plan or a blockchain agnostic application this could be avoided but leads to, again, a lot of overhead. A blockchain agnostic application is distributed over multiple blockchains instead of just one. Research projects exists in terms of agnostic blockchains but currently agnostic applications are dependent on an trusted institute which executes the trades between the blockchains.

An interesting problem of complete anonymous and private transactions would be the potential exploit of the soundness property of the zero knowledge zk-SNARKs. Transactions which are invalid but counted as valid could be created. This means that the transactions are seen as valid and can generate cryptocurrencies and value out of nowhere when using it in a cryptocurrency application. The generated transactions and cryptocurrencies are valid according to the network even though they are only exploiting the application. This of course is bad due to the hidden inflation. It is hidden because it is not publicly visible what is happening behind the transactions.

Even though Z-Cash is the first widespread application of zk-SNARKs [30], Monero is using the Bulletproof technology and QED-it is only using it in a theoretical way I think zk-SNARKs can be the way to go if a trusted instance exists. The three applications

are fundamentally different but display the possibilites with Zero Knowledge Proofs, Bulletproofs and zk-SNARKS. Also the evolvement of STARKs and Bulletproofs show that there is interest in further developing the technology. So Zero Knwoledge Proofs and applications are coming but it is yet to decide which technology establishes as the most practical, In my opinion it will always be a matter of the use case.

Chapter 4

Problem Analysis

As already stated in Section 1.2 there are 3rd party companies which are trusted by institutions. For example tax agencies or cloud services which calculate realized gains of cryptocurrency trades are trusted by the government to provide the correct (not modified) tax result. If a user wants to use this offer one has to give away data which is needed for example the calculation.

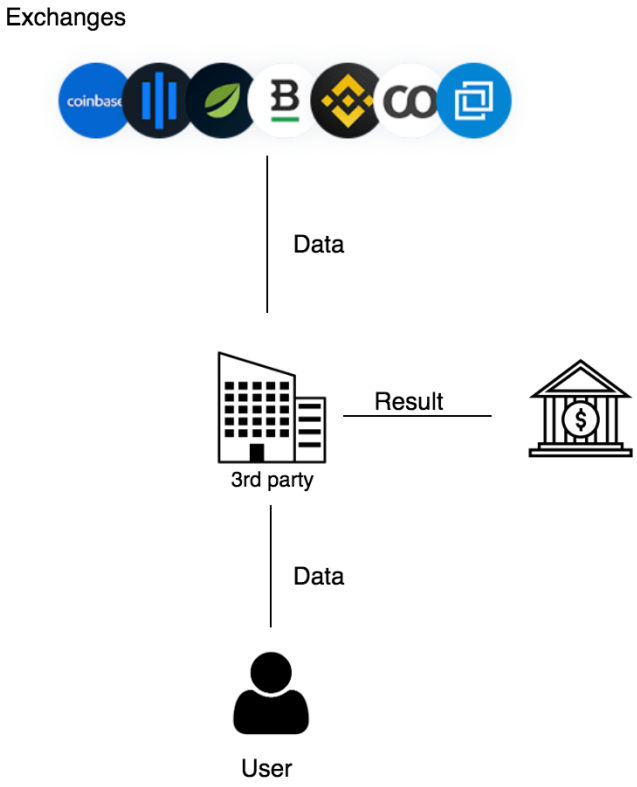


Figure 4.1: Example of how a Saas (3rd party) structure can look like.

As visualized in Figure 4.1 there are a lot of 3rd party SaaS (Software as a Service) companies for which this is true. More and more services are cloud based due to a lot of advantages and security reasons. The 3rd party company maybe uses the data to make calculations or access other services to make the life of the user easier and provides a benefit or result for the user in most cases. This result is, because the company is trusted, accepted at other institutions. However, since a lot of user use this approach the 3rd party company collects a lot of data which can lead to problems. This can range from unwanted hacks to leaks of the data. Since most of the data is centralized within a database there exists a single point of failure.

Because of the new EU General Data Protection Regulation (GDPR) introduced in 2018, companies also have to implement privacy by design as a default mechanism. Products and the companies themself should comply with the seven principles of privacy by design as stated in Section 2.4. There are already approaches for how to implement this in companies. However when designing a decentralized application and evaluating it with the privacy by design this might be another challenge.

It is yet unknown how a zero knowledge validation within a decentralized application with multiple data provider works. This needs to be evaluated within a prototype but also which parts could be used within a real live application. It should also be defined with which principles of privacy by design such a system complies and what needs to be changed, if it can be changed, to comply with the principle.

Chapter 5

Methodology

The implementation of the prototype and its evaluation is divided in four parts which are visualized in Figure 5.1 as the project process. The Research Analysis is already explained in Section 3 and defines the basis for the following steps. The methodology for the architectural design will be explained in the following Chapter whereas the implementation will be explained in Chapter 6. The final analysis of the prototype as well as the evaluation will be discussed in Chapter 7.

The following sections are about the concepts used for the prototype as well as how the prototype will be evaluated against the principles of privacy by design.

5.1 Concepts for the Prototype

To identify which concepts are needed for the prototype, a simplified approach on how a solution can be designed as visualized in Figure 5.2.

The shown figure is a decentralized version of the well known approach discussed in Chapter 4. By removing the 3rd party and adding a public blockchain which is accessible for every instance within the system a solution with no major function drawback could be accomplished. The following chapter describes the methodology used to build a prototype and how it is validated against the principles of privacy by design.

5.1.1 Swappable Blockchain

To have a persistent and decentralized storage which is not controlled by a individual instance a Blockchain based storage is chosen. Parts of the application can be built on top of a existing blockchain infrastructure to avoid maintenance and overhead. For the prototype this should be only running local but in a way in which the blockchain can be

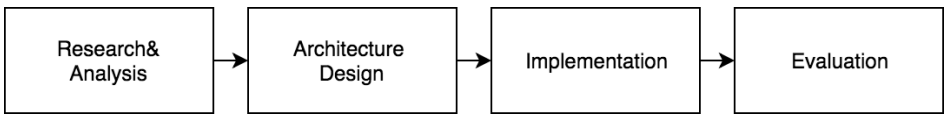


Figure 5.1: Process of the prototype design, implementation and evaluation.

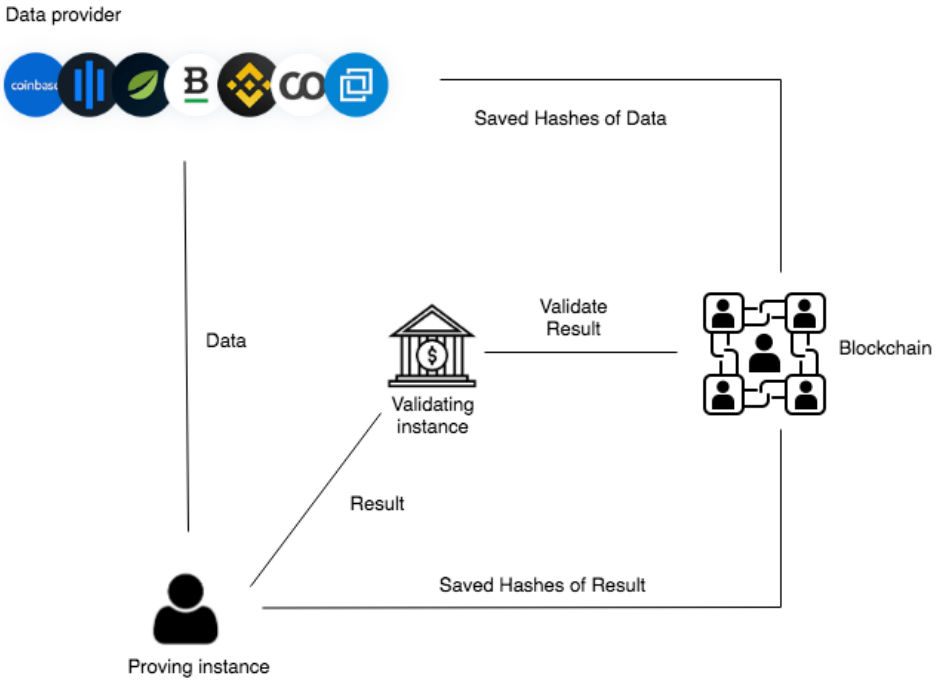


Figure 5.2: Simplified visualization of how a solution can be designed.

swapped to another blockchain at any time. For this the application needs to be written in a way that it can easily be changed. The concept of the adapter pattern should be applied to design the interface in a way to be easily changeable or swappable. With this there might be also the possibility to choose in which blockchain the results or data should be stored if a blockchain is preferred. For the prototype however there should be only one blockchain which is running locally and non distributed.

A existing public Blockchain is chosen to make it available for every instance in the system. With a proper provided API of the application every instance can read and write to the blockchain. The manipulation, as already mentioned in Section 2.1, is very hard or even near impossible due to the big distribution.

5.1.2 Anonymous User Identification

When interacting with the data provider and storing data on the blockchain it must not be visible to any instance (except the validating instance) to which user data belongs to. For this the validating instance should generate a databox for the user and initialize it on the blockchain. The databox stores all the validation hashes and mechanisms for this request lifecycle. The lifecycle starts when the user requests the databox from the validation instance and ends when the proof of the proofing instance is validated by the validating instance. If the user wants to restart or aborts the proof lifecycle due to an error or something similar, the databox must not be able to be connected to the old databox. It cannot be destroyed due to it is stored on the blockchain. The linking

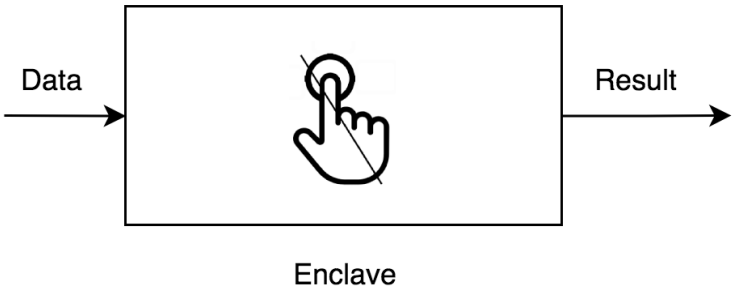


Figure 5.3: Calculation is happening within the enclave.

from databoxes to the user must not be possible for instances which do not have the permission to do this. Only the validating instance knows which databox is linked to which user. The user itself and every data providing instance are only working with the databox to store and read data from it.

To have the possibility to create proofs which are only be able to be validated by the validating instance a encryption key pair should be generated. Only the validating instance is in possession of the private key whereas the public key is used by the other instances. This also needs to be unique for every validation lifecycle.

5.1.3 Calculation in simulated Private Protected Area

The protected area (or enclave) is an area which is running on the users device but closed to external intrusion. The data gathering as well as the calculation should be performed within this area. This process is important to make sure that a user cannot manipulate the fetched data or make adaptions to the calculation itself. This is ajared to the concept of QED-it used in Section 3.1.2. For the prototype this should only be simulated to have a proper validation later. The enclave as displayed in Figure 5.3 is a non interaction area in which only data is inserted and a result received. The enclave is, in theory, manipulation safe.

5.1.4 Calculation within Blockchain Application

As an addendum to Section 5.1.3 a possibility to be able to calculate the result on the blockchain itself should also be possible as visualized in Figure 5.4. This might not be a use case used in an live environment but should reveal problems and serve as an additional parameter for the privacy by design evaluation.

The use of the blockchain for to the calculation itself can lead to higher transparency due to the fact that the algorithm of the calculation is publicly visible on the blockchain. The enclave operates in the same way as in Section 5.3 to minimize the possibility of bad external influence.

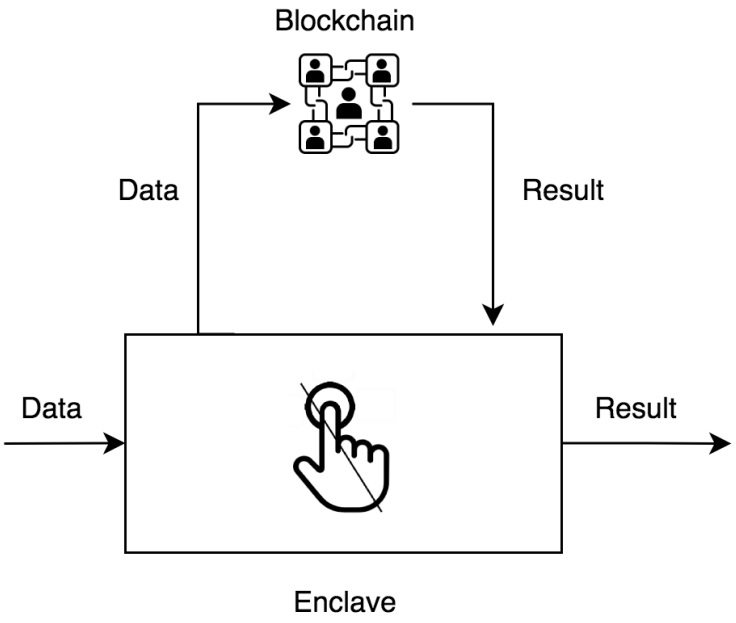


Figure 5.4: Calculation with the enclave in combination with a Blockchain.

5.1.5 Data Grounding

Every piece of data which is needed to generate proof or validate proof needs to be grounded on the blockchain. This needs to happen in a way in which the original data cannot be revealed by any instance at any time but still be available for every instance to validate it. The grounding algorithm needs to be the predefined and publicly available for every entity to generate proofs which can be validated later.

This can be achieved by hashing parts of the data in a way the proving or validating instance can validate it. However the validating instance needs the data in different way than the proving instance to recreate and check the proof. For this a mix of hashing and encrypting of the data should give enough flexibility. This however will generate a weak spot when the private key used to decrypt the data is exposed.

By defining rules on how the data is hashed and which data gets encrypted the generated proof is the same every time. To achieve the desired result the data generated by the data provider as well as the generated proof of the user needs to be grounded on the blockchain.

5.1.6 Commitment Calidation

To be able to validate the data received from the data providing instances and check if the proofs generated by the other instances are valid, a proofing mechanism needs to be generated. Because every data provider and other instance ground the proofs on the blockchain as stated in Section 5.1.5 the proofs are available for every other instance to validate. This validation could be for example triggered directly in the enclave to

check if the data received from the data provider is the same as the one grounded on the blockchain.

The validation itself needs to be done within the blockchain application. This should be realized similar to the concept of the enclave. When validating a proof via the provided API only true or false will be returned. No additional information is revealed due to the fact that this information could lead to the reconstruction or possibility to fake the result. The proof however needs to be generated via the instances to avoid sending data unnecessary. The algorithm how this is generated is already stated in Section 5.1.5.

The commitment validation will also be the the major part of the zero knowledge section. To check if it is theoretically possible to validate the proof and every other data without revealing which data was used to generate the proof.

5.2 Privacy by Design Validation

The prototype which uses the concepts explained in Section 5.1 should be evaluated. The result is an evaluation if such a decentralized application can comply with the concepts of privacy by design. The concepts as described in Section 2.4 are rewritten to defined tasks which need to be fulfilled. The aspect privacy by design can only be fulfilled if every aspect itself is fulfilled. The following paragraphs define under which circumstances a principle is considered as fulfilled for this application.

Proactive not Reactive; Preventative not Remedial

- The architecture is designed to prevent privacy invasive events.
- The system allows continuous improvement as well as the correction of errors.

Privacy as the Default

- All personal data is automatically protected in every step processed.
- The purpose why data is collected and used is communicated to the user.
- Only the data which is necessary for the specified purpose is collected.
- The link ability to personal information is minimized or non existent.

Privacy Embedded into Design

- Privacy by design is a core functionality delivered.
- Due to this being a prototype no stakeholders and interests need to be consulted.

Full Functionality – Positive-Sum, not Zero-Sum

- The decentralized solution has to have the same functionality as the one which would be offered by the 3rd party company.

End-to-End Security – Lifecycle Protection

- Data is transferred securely through the entire lifecycle.
- If data is not needed anymore it gets destroyed.

Visibility and Transparency

- When transferring private data to third parties the data is secured.
- Policies and practises relating to the management of personal data is communicated in a open and transparent way.
- Steps to monitor, evaluate and verify compliance are taken.
- Complaint and redress mechanism are available.

Respect for User Privacy

- Individuals have access to their data.
- Individuals are able to challenge the accuracy and completeness at any time.

The prototype is not only validated against the seven principles but also compared how the prototype complies when using it within the test but also a live environment.

Chapter 6

Solution Approach

Based on the problem stated in Chapter 4 a prototype is developed. The implementation approach is based on the concepts described in Section 5.1. The following chapter will explain the implementation of the prototype in detail and gives the foundation for the evaluation with the principles of privacy by design.

6.1 Requirements

Even though the application to generate the proof is running on the users device to keep data decentralized, the application needs to be developed by a trusted entity. This may be a trusted software provider which develops, releases and audits the software. Such a step is necessary in a live environment to have the trust and validity that the software is working correctly as intended. If such an audit is not given no one made sure that the process is working correctly.

A software setup as displayed in Figure 6.1 would enable that the user gets the updated version of the software if previous versions might contain bugs because of the maintainer. This is also interesting if for example regularities change or new laws apply when using a tax software. The software provider could of course also be the validating instance.

For this prototype it is assumed that every instance in the environment is open to implement the solution. This means that the data provider enable the possibility to save data on the blockchain. It is also necessary that the instances communicate in an

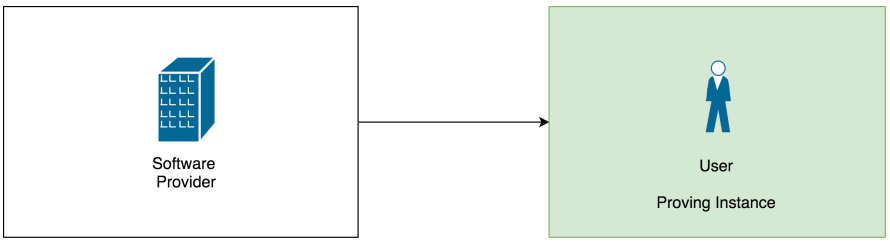


Figure 6.1: A software provider shares the audited software with the proofing instance.

encrypted way with each other. The proving instance also needs the perfect hardware setup which supports for example the use of enclaves. The validation instance which might be a public instance such as a government in the real environment needs this such a hardware setup as well.

6.2 Architecure

The architecture of the prototype itself is divided into five main parts which are described as the following and visualized in Figure 6.2. The main requirement is that the situation stated in Section 6.1 needs to be fulfilled.

- The Software Provider who is already described in the Section 6.1 has the task to develop the software application. The validation instance needs to accept the software provided in a live environment. To have a real transparent application the code should be hosted open source. The software provider could theoretically be also the validation instance. However to have an independent developed application also has benefits when talking about a decentralized architecture.
- Data Providers (or exchanges) which are represented within the white box provide the data which is used by the proving instance to execute the calculation. Those data providers should never communicate with the validation instance. The data stored by each data provider must not be able to be revealed by any instance. Only the proving instance has access to the provided data.
- The Validation Instance is displayed as the blue box and is the trusted instance withing the prototype environment. This instance needs to trust the software and takes part in it. This instance needs to communicate with the user as well as the Blockchain to validate the proofs generated by the user. It is also the only instance which is allowed to initialize a proofing lifecycle.
- The Proving Instance is in the green box and represents the user which is using the software provided by the Software Provider. This allows the user to calculate results and generate proofs with the software provided by the Software Provider and the data provided by the data providing instances. The user itself who is the proofing instance needs to proof that the submitted result is valid.
- The Blockchain Instance is in the yellow box and is the public transparent data storage for all instances. The decentralized storage which stores all proofs allows every other instance to access it. This is possible with a provided API allows all other instances to make the same requests which can be read and write requests.

Figure 6.2 visualizes the steps necessary for one validation request which is one request lifecycle. Those necessary steps are described in the following paragraphs. The ways of how the validations and proofs are generated and validated are described later in Section 6.3

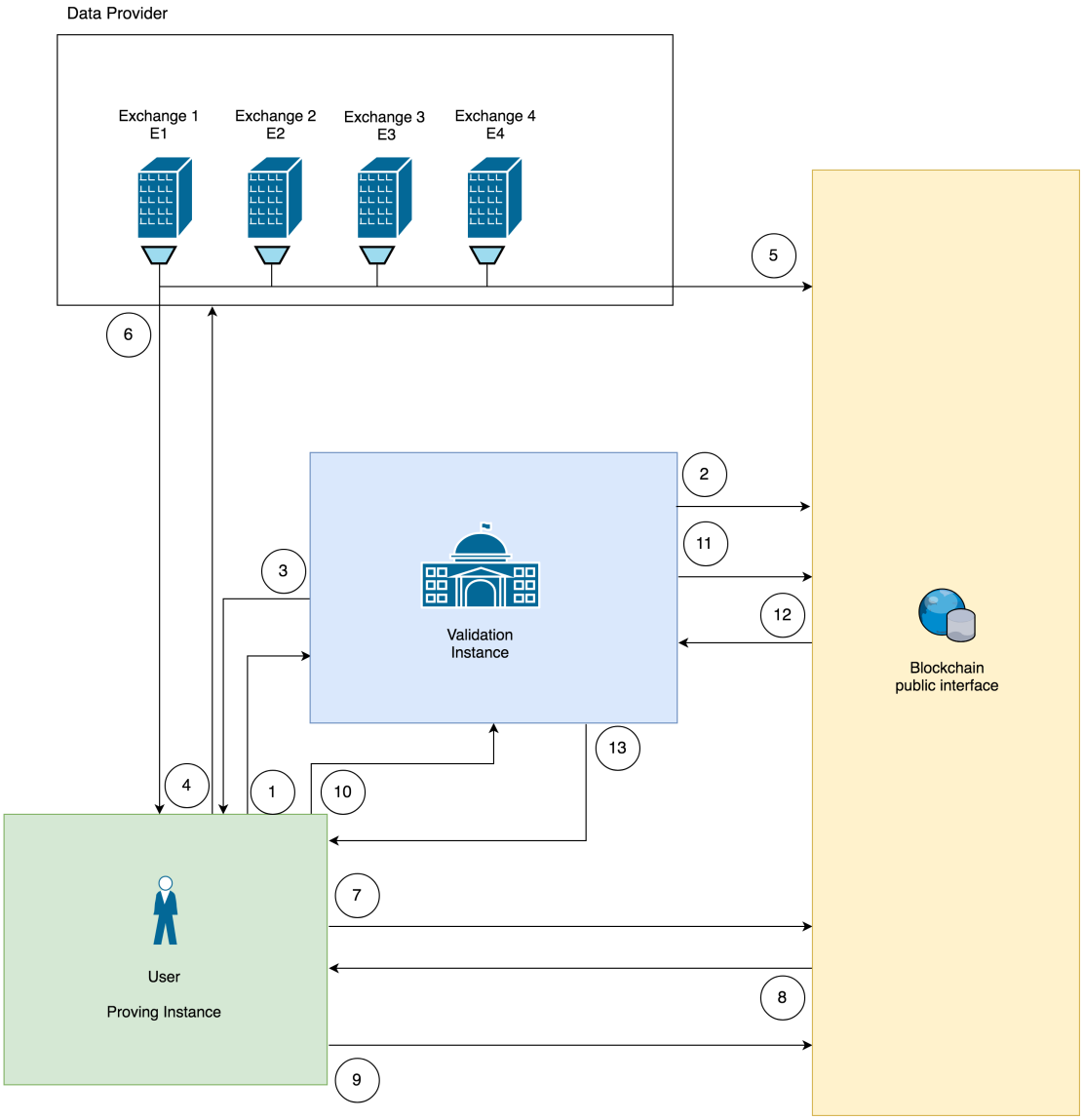


Figure 6.2: The full architecture of the prototype environment.

1. The proving instance is required to initialize the request lifecycle. To achieve this a request with the unique identifier of the user is sent to the validation instance. This unique identifier is the authentication of the user.
2. The validation instance accepts the request of the user and initializes a databox on the blockchain for the user. This databox is later used to connect all proofs for this request to the users request. The request on the blockchain is instantiated via the API provided by the blockchain application. Also an encryption pair is generated for later requests. The private key is kept at the validation instance but the public key is shared with the user.

3. The validating instance communicates the location of the databox to the user as well as the public key. With this information the user is now able to connect to the data provider.
4. The proving instance needs to fetch the data from the data providing instances. For this a request is sent to every exchange. The public key as well as the databox location is sent to give the exchanges the information needed to ground the data.
5. The exchange receives the request and validates it. A hash of the data is generated and stored encrypted on the blockchain. This is later used by the validation instance to generate the proof. A hashed version of the hash is also stored on the blockchain for the user to validate if the same data is used by the exchange and proving instance.
6. The data provider returns the data to the proving instance.
7. With the data received from the data providing instance a validation check is performed. The data is double hashed and sent to the blockchain application. Not only the similarity of the data checked but also the amount of the data providing instances used. Both the content and the amount have to match with the values stored on the blockchain.
8. The blockchain application returns either $true$ or $false$ depending on if the received data is the same as the data the data providing instance worked with.
9. After the data is validated to be exactly the same as the one from the data providing instance the calculation can be performed. With the result as well as the data from the data providing instances the proof is generated. This proof is stored on the blockchain for later validations.
10. The calculation result, the generated result proof as well as the hashes of the exchanges are submitted to the validation instance. This allows the validation instance to also generate the proofs.
11. With the received data and the encrypted hashes stored by the data providing instances on the blockchain the proof can be validated. For this the validation instance sends all data received from the user to the provided API endpoint of the blockchain application.
12. Depending on if the data received from the user is valid the blockchain application either returns $true$ or $false$.
13. The validation instance sends the proving instance either $true$ or $false$ depending on if the provided data is accepted or not.

6.3 Validation and Proof Generation

The following section will give a general overview on how the proofs are generated as well as validated. This is displayed in a general way to keep the possibility to implement this in any system. The parameters themselves are explained on their first occurrence.

6.3.1 Initializing the Request Lifecycle

When initializing the request lifecycle on the blockchain a *databox* for the user is generated. This *databox* contains all data which is related to the user. Data of exchanges

for example are accessed via

$$databox_{exchanges}. \tag{6.1}$$

The other main part which is generated by the generator g at this point is the public and private key as displayed in Equation 6.2. An asynchronous encryption is recommended due to the fact that the guessing of the encrypted value is way harder. λ in this function represents a random unique identifier which is only known by the validation institution to generate randomness in the keypair via

$$(privK, pubK) = g(\lambda). \tag{6.2}$$

6.3.2 Grounding Proof of Data Provider

Grounding the data of the data provider is an essential part to be able to generate proofs later. Withing this the data provider are represented as E. Due to the fact that multiple data provider exists they need to be accessed with an index E_i. The data set of a data provider is represented as tx and is a collection of transactions. Those transactions must also be accessed with an index such as tx_j. Looping through all transactions of an exchange would range from j which represents the first transaction of an exchange to n which is the last one. With this information it is possible to generate a hash for each Exchange which is represented as E_ith as

$$E_ith = hash(\sum_{j}^{n} E_itx_j). \tag{6.3}$$

With this Exchange Hash E_ith the proof for the validating instance and the proving instance can be generated. The proof for the validating instance can simply be generated by encrypting the Exchange Hash E_ith with the public key $pubK$ received from the proving instance. The resulted E_ithe represents the encrypted value of the Exchange Hash and is generated with

$$E_ithe = encrypt(E_ith, pubK). \tag{6.4}$$

The proof for the user is generated as

$$E_ithh = hash(E_ith). \tag{6.5}$$

For this the Exchange transaction Hash from Equation 6.3 is simply hashed again. The simple Exchange Hash E_ith is never stored anywhere in clear for the proving instance. However the generated double hashed value E_ithh can be stored without any problem. Both generated values E_ithh and E_ithe are stored in the *databox* of the user for later validation. The E_itx is also available for the proving instance to work with.

6.3.3 Validating Data Provider Input

To validate if the proving instance is working with the same data as the data providing instance two checks are performed. The first check validates if the amount of exchanged received is the same as the one stored on the blockchain. The check

$$\sum Exchanges = \sum databox_{exchanges} \tag{6.6}$$

is necessary to perform the second check.

Because the proving instance receives the E_itx it can also generate E_ith by using the same method as in Equation 6.3. After generating this a check is performed if every E_ith generated is stored on the blockchain. This is done by hashing the E_ith generated by the proofing instance to generate E_ithh which was previously grounded by the data provider in Section 6.3.2 with

$$\forall i \exists x; x \, \epsilon \, databox_{exchange} \wedge x = hash(E_ith). \tag{6.7}$$

If both checks are valid the proving instance can be sure that the data used is the same as the one used by the data providing instances and also later by the validating instance.

6.3.4 Generate Calculation Proof

After having validated the received data E_itx in Section 6.3.3 the proof can be generated by using E_ith in combination with the *calcResult*. The *calcResult* in the prototype is the result of a simple addition of the received data but can theoretically be anything. The application running within the enclave is handling the *calcResult* which is the output when inserting the E_itx of all Exchanges. The $calcHashuser$ is generated by adding all hashes of the exchanges to the generated *calcResult* with

$$calcHashUser = hash(\sum_{i}^{n} E_ith + calcResult). \tag{6.8}$$

6.3.5 Validate Calculation Proof

The requirement to have a proper last validation step is a legitimate generation and submission of the *calcResult* as well as the order of the E_ith used. The validating instance which has to validate the transactions is not in the possession of the E_itx. However it is in the posession of the private key $privK$ which was generated in the first step as explained in Section 6.3.1. Because of the fact that the data providing instances grounded the Exchange Hashes E_ith in Section 6.3.2 encrypted as E_ithe on the blockchain the validating instance is able to work with them and generate E_ith again with

$$calcHashValidator = hash(\sum_{i}^{n} decrypt(E_ithe, privK) + calcResult). \tag{6.9}$$

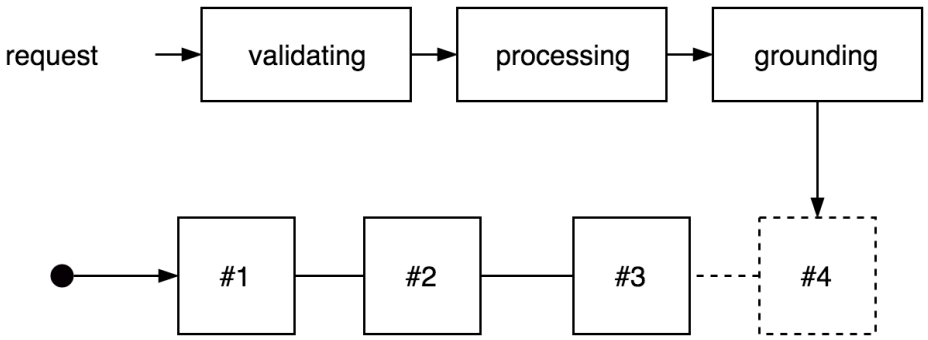


Figure 6.3: Process to ground data on the blockchain within a new block.

The condition is that the proving instance submits the *calcResult*. With all information gathered it is possible to generate the proof of the validation instance $calcHashValidator$. The proof is considered as valid if the $calcHashUser$ of Section 6.3.4 is exactly the same as $calcHashValidator$.

6.4 Implementation

The implementation as explained in the following chapter is crucial on how the prototype is realized. Having a general look the architecture is designed to have every instance as an independent server instance which are all realized as node.js applications. The instances must communicate only via a REST API by sending json objects containing the required information each server needs or provides. The REST API is the part of the server which is publicly available to the other instances.

The hashing algorithm used doesn't matter for this prototype if every instance knows the rules and is using the same algorithm. However when using this in a live environment one might consider using a collision safe or quantum resistant hashing algorithm. To have an easier use of the prototype the proving instance as well as the validating instance provide a web interface. This interface should be used to generate the proofs and it offers a simple possibility to valid the proof sent from the proving instance.

6.4.1 Blockchain Application

The Blockchain is not realized as a real decentralized application but rater a simulated decentralized Blockchain due to development reasons. In a Blockchain, such as Ethereum, which would be used in a live environment a time factor determines when a Block is appended to the Blockchain. It is also possible that the Block is full and the data which should be grounded on the blockchain is not grounded in the next block but in the one after that. Because of this a JavaScript based Blockchain was chosen to have more development possibilities and quicker development cycles. Theoretically the JavaScript Blockchain could be swapped with any other Blockchain which has the application for this prototype written on top of it.

The JavaScript based Blockchain allows to develop and make changes fast and easy.

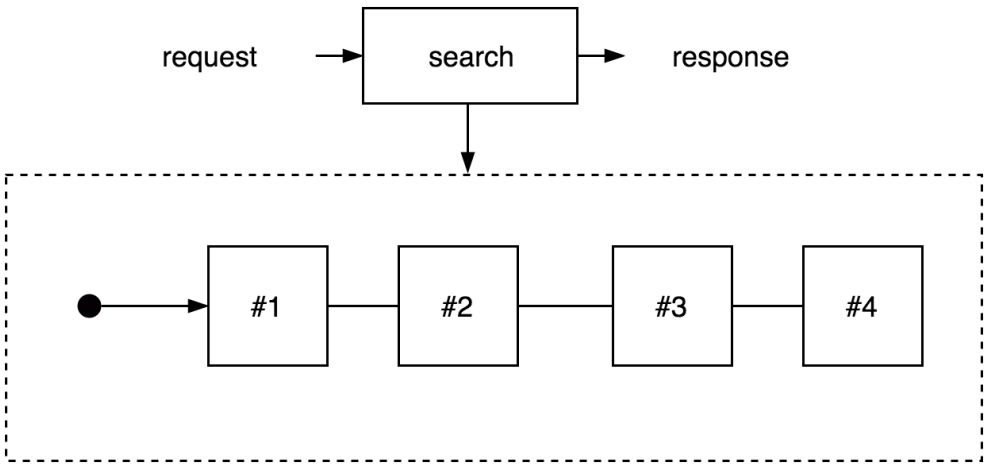


Figure 6.4: Process to read data from the blockchain.

As Figure 6.3 shows the process when grounding the data on the blockchain is divided into three parts. The request which is coming from the API is validated in the first step to be sure that the right data for the request is sent. This also contains a white list check to validate if the requester is authorized to perform this action. The proving instance for example can trigger different API endpoints than the data providing instance. If this check is considered as valid the data is processed, normalized and finally grounded on the blockchain. In the case of this prototype the JavaScript based Blockchain generates a new block for every write request. This means that every block contains one grounding request.

Reading data from the Blockchain is a rather simple process compared to the grounding of the data as Figure 6.4 shows. Requesting data triggers a complete search through the whole blockchain. This means if for example the amount of exchanges and their data should be validated, every block is read and checked if the data of the user is stored in there. Having used this process in the development environment caused no problems but it might be a problem when using this in an live environment with millions of blocks. However another Blockchain would also store more transactions within one block which would also reduce the amount of blocks. The response of the request returns either data or just a `true` or `false` statement depending of the requested API endpoint.

6.4.2 Data providing Instance

This prototype utilizes three data providing instances but could be extended to a finite amount depending on the power of the used devices. The provider has the simple task to generate data for the proving instance and ground it on the Blockchain in a way it cannot be reverted to it's original values but still be validated. For this prototype the amount of data as well as the values are generated randomly to have different data for each request lifecycle.

Each data providing instance loops over a dataset which Program 6.1 shows and generates the hash with the rules stated in Section 6.3.2. The instance ends up with a

Program 6.1: Example of a dataset provided by a providing instance.

```
[
  {
    "id": "694-Exchange2",
    "value": "5115"
  },
  {
    "id": "277-Exchange2",
    "value": "629"
  },
  {
    "id": "988-Exchange2",
    "value": "-8256"
  }
]
```

Program 6.2: Example dataset of the grounded data of an instance.

```
{
  "storage": "52a79ae023fccec3e9b9fd257898bdd5268659f5cc0b7010efb85a3adfa87f22",
  "value": "T6hwfbDifTx3zVtIp5QXgs8iimHsv8wvxs6yPBpSlVtCwaSKYouiZ+
            GWogEkF1DH8v1xO8skweiAtZjwib0pzvkdaaVEwp486EZPdGXssSop0pvCq2XZls
            objeIR3NJqudSYkWjj/omohGYn9Gr71h61VCsogeaJT6eTeD/lVp09wwymVcH12s
            GR8C8tFcL2FVbvWluWjvHonpNgsa7QA79n98FCv8bNGw0knSUoqgmu9dL4oMXKI3
            mgvSOShIIZ9wsuKTmkA9ef5mnGAo4j87prCHkguOKOWkyzZsXSYFIwtAfzXjI6jo
            9nhzeTy4tXw2O3obsgrphGWNCi08Mawg==",
  "hash": "f3e06b6de6cf905ae411c32a1ba2e1f3bf90c68dc18211b78d"
}
```

proof grounded on the blockchain in which the `value` is the encrypted hash and `hash` and the double hashed value as demonstrated in Program 6.2.

The `storage` in the grounded data is the location of the databox of the proving instance. This is performed for every data providing instance and generates the desired proofs on the blockchain. The `value` is later decrypted and used by the validating instance whereas the `hash` is used by the proving instance to check whether the received data matches or not.

6.4.3 Proving Instance

The proving instance, or in this case the user, is the instance which is connected to every other instance. After initializing the request lifecycle with the validation all data from the data providing instances is fetched as described in Section 6.4.2, the data fetching, validation and calculation is executed.

The datasets received from the data providing instance are validated within the web interface by generating the same proof which is stored on the blockchain. The provided web interface displays the validation of the proofs as displayed in Figure 6.5 and shows if all received data is valid. If data is manipulated or different than the fetched data, a

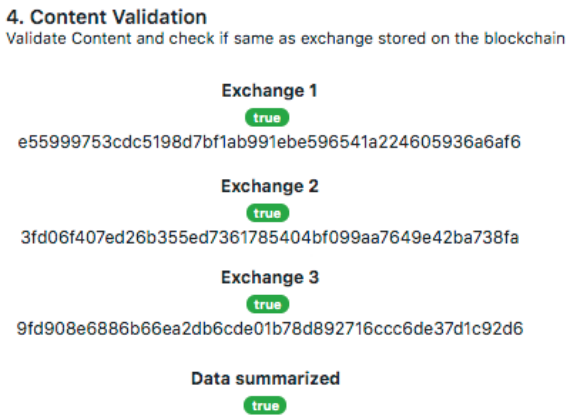


Figure 6.5: Validation of the received data in the prototype interface.

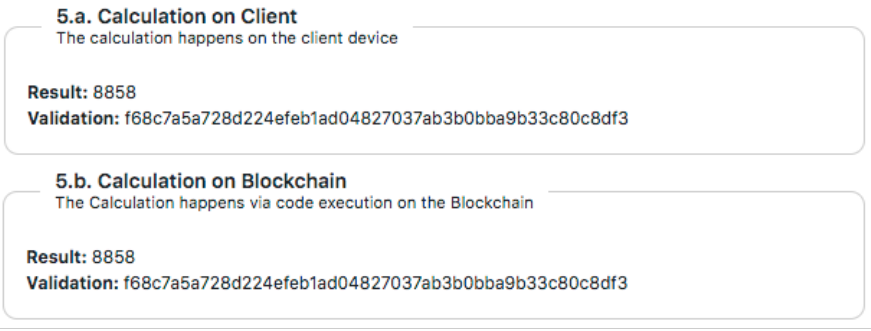


Figure 6.6: Displayed result of the calculation on the Client and Blockchain.

`false` signals the user that the data is faulty and no further calculation is possible.

With a valid data set of all data providing instances it is possible to continue with the calculation. To have a selection and comparison on how the result will be calculated, two different methods are offered as visualized in Figure 6.6. The first method calculates the result within an enclave. The enclave as explained in Section 5.1.3 calculates the data in an environment with which the user cannot interact with. The manipulation of the data is not possible. The second approach is to have the calculation within the Blockchain application. The logic is the same as in the enclave but all datasets are sent to the Blockchain application to be calculated and a result is returned. The dataset is also validated within the blockchain to make sure that the data is exactly the same.

The final object as stated in Program 6.3 is an example of the content submitted to the validating instance as last step for the proving instance. The `resultHash` is the final proof containing the `result` of the calculation as well as the exchanges generated with the rules stated in Section 6.3.4. The `exchanges` are necessary to maintain the same order when the validating instance is generating the `resultHash` to proof it.

Program 6.3: Full proof which is later validated by the validating instance.

```
{
  "uuid": "708392fc-c82f-33f7-89e1-554498606373",
  "result": "23825",
  "resultHash": "ca373ee5686be016dcf2b2864fd3360363e9c5312108d4b6e5",
  "exchanges": [
    "981f9a6b01c4615fdf7a1735b6a21920fd69fc481d21066ba5",
    "400219d91fa21d212cd68fc60153e3bf9915eb347d02de3faa",
    "c54553b426335eb9380e9c757aa367430d0e9995286e592a37"
  ],
  "storageLocation": "16483
    cf42347751b472e12d5bd33cd207705e45c0bcefd57db08b58f46268c16"
}
```

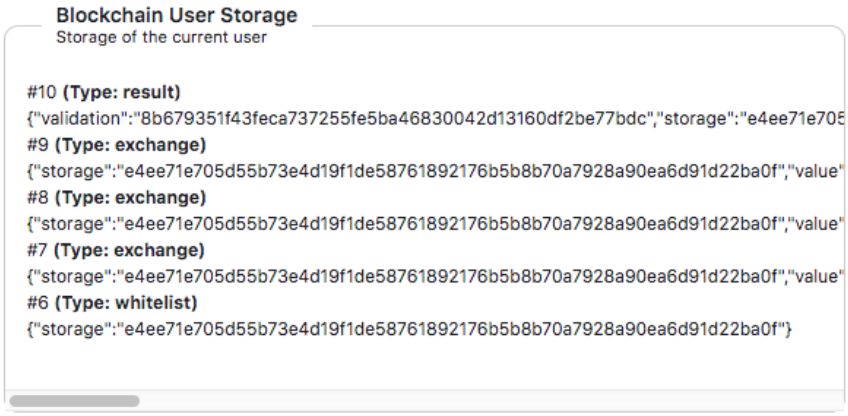


Figure 6.7: Displayed result of the calculation on the Client and Blockchain.

To keep transparency during the whole process the storage of the user on the blockchain is always visible for the user. With the blockchain monitor as displayed in Figure 6.7 the user can monitor what data is processed and stored in real time.

To have complete transparency not only on the user data but on all data processed within the whole blockchain a second monitor allows the user to detect any changes as displayed in Figure 6.8.

6.4.4 Validating Instance

The validating instance is the first but also the last instance within an request lifecycle. As first step the initialisation of the databox or storage location is triggered as well as the keypair is generated which was already stated in Section 6.3.1. The information of Program 6.4, which contains the public key `key` and the location of the databox `storageLocation`, is send to the proving instance to work with.

After the proving instance generated the proof, the validating instance receives the data which is mentioned in Section 6.4.3. The initial task for the validating instance

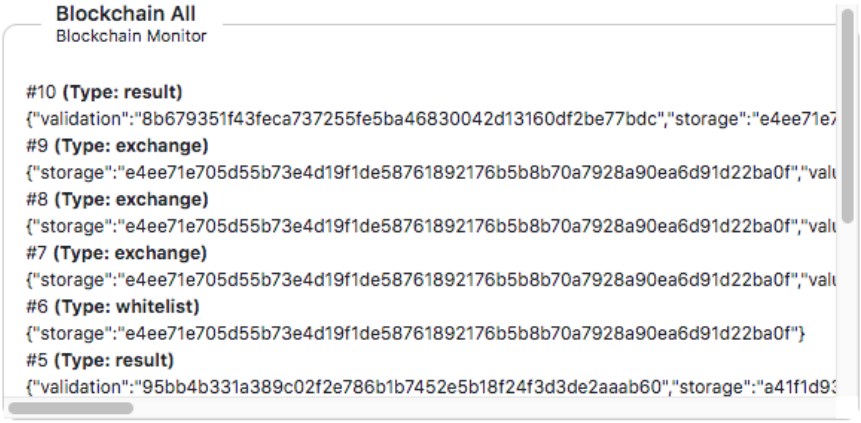


Figure 6.8: Displayed result of the calculation on the Client and Blockchain.

Program 6.4: The initial data sent to the proving instance.

```
{
  "key": "-----BEGIN RSA PUBLICKEY-----
          -- the rsa public key would be here
          -----END RSA PUBLIC KEY-----",
  "storageLocation": "
    fb5a25f8a26fe88fd0687b4efe11fe4fe6e51bc4f0669b8fe6caf73bfab3be91"
}
```

before validating the proof is to check if the `storageLocation` as well as the sent `uuid` of the proving instance is valid. After this a check if performed to validate if the exchange hashes receives match the ones stored on the Blockchain. For this the validating instance received all encrypted data providing hashes which are grounded on the Blockchain as explained in Section 6.4.2.

The web interface of the validating instance accepts the data generated by the proving instance and displays the proof as seen in Figure 6.9. Because this instance is in the possession of the private key of the proving instance, which can be matched to the received uuid, the validating instance can decrypt the encrypted hashes on the Blockchain and check if they match with the ones provided by the user which is displayed in Figure 6.10. If this is the case the validation proof is generated with the rules explained in Section 6.3.5. The generated proof must be exactly the same as the one received from the proving instance.

Figure 6.9: Display of the submitted data in the web interface.

Figure 6.10: Display of the final proof in the web interface.

Chapter 7

Evaluation

The prototype implemented in Chapter 6 is used to validate if all Zero Knowledge Properties are fulfilled. The following chapters discuss the evaluation of which principles of privacy by design the prototype is fully, partial or not compliant with the prototype to be able to answer the research question. Potential issues and conflicts are also reviewed to be able to have further research and improvement of the prototype.

7.1 Zero Knowledge Proof Validation

A major requirement to validate the prototype against the principles of privacy by design and discuss the research question as introduced in Section 1.3 is that the Completeness, Soundness and Zero Knowledgeness properties of Section 2.2 are fulfilled. The following chapters define how the properties are fulfilled.

7.1.1 Completeness

The formal language L is defined dynamically for every user as dataset which contains among others the hashes generated by the data providing instances. With a containment check if a hash x is part of L it can be ensured that only valid data is taken within the calculation. This is achieved with the check of the generated hashes against the data received from the blockchain.

When validating the calculated result the formal language L consists of only one element: the generated proof of the proving instance. The proof is valid and the validating instance is convinced that the `resultHash` submitted by the user can be generated with the data received from blockchain in combination with the submitted result.

7.1.2 Soundness

To fulfill the Soundness property the system instantly aborts processes if validations fail or are considered as not valid. Because of this not only the elements needs to be part of the Lanugage L but also the amount of the elements need to match.

Having the generated `resultHash` to be exactly the same as the one submitted by the proving instance is also necessary. The proving instance makes the same checks as the validating instance and instantly aborts if failures occur. Because the saved hashes

of the data providing instances are only hashed the proving instance might be able to guess which value is hidden behind the hashed element and fake the result. Even though the chance is very low this is theoretically possible.

7.1.3 Zero Knowledgeness

The Zero Knowledgeness property is fulfilled by only sending hashes and encrypted values of the hashes to the blockchain to generate the proofs. The data itself is only transferred between the instances which need it to generate the initial proof. The data providing instance is only sharing the data with the proving instance while the proving instance is only sharing data with the validating instance. Because of a secure transfer no man in the middle attack should happen. The validating instance as well as the blockchain is not able to reconstruct the full data with the hashes.

The generation of the proof itself can be recreated by the validating instance without the need of giving the clear data to the validating instance. The proof can be recreated with the encrypted hashes stored on the blockchain. Those are only accessable by the validating instance which is in the possession of the private key.

A user is not connectable to a databox because for every new request lifecycle a completely new databox is generated. The validating instance is the only instance which can connect the databox to a user. The databoxes are not be connectable to other databoxes.

7.1.4 System Classification

The system can be considered as a perfect zero knowledge system due to the fact, that a single request with the proofing data can convince the validating instance that the submitted result is valid or invalid. Multiple requests which are needed for a statistical Zero knowledge system to convince the validating instance are not necessary. This would also be working within this system due to the fact that a perfect proof is necessary and not a statistically close proof. The simulator S can generate the `resultHash` at any time with the given data and reproduce the proof perfectly every time the same data is inserted. When comparing the submitted proof and the generated `resultHash` they must be exactly the same every time it is submitted.

7.2 Privacy by Design

To evaluate the principles of privacy by design the architecture and method of the operating of the prototype as introduced in Chapter 6 is analysed. Every principle is analysed on its own as well as summarized later for the calculation within the enclave and within the blockchain. Issues and conflicts which may occur are discussed in Section 7.3.

7.2.1 Principles Evaluation

The following principles are evaluated as described in Section 5.2. Every principle is analysed, discussed and explained in detail independently. This is done for the prototype

which has been introduced in Chapter 6 but also, if necessary, as contrast, for the application within a potential live environment.

Proactive not Reactive; Preventative not Remedial

The architecture of the prototype definitely fulfills the fact, that it is designed in a way in which privacy invasive events are prevented. This is on the one hand the fact because every user does not have to give away data and stores it on the own device. On the other hand the possibility to calculate everything within the enclave. However the continuous improvement of the applications is only partly possible.

The improvement of the application depends a lot on how the application itself is developed. With the approach of having a software provider, the software provider can provide updates and update the application on the validating, proving and data providing instance. However the blockchain application is, once it is deployed and running, not modifiable in this prototype. This also includes data which is stored on the blockchain.

Within a live environment there are multiple ways to update a blockchain application. This may be however not a simple update but a new application which copies a snapshot of the data from the old application. The old and maybe faulty application is still on the blockchain and cannot be removed. This is for example the case when developing the application on the Ethereum blockchain.

Privacy as the Default

Privacy is a major part in this Zero Knowledge System. The default mechanism is that most of the application is running on the users device itself. Because of this the user does not have to give data to third party companies in the first place. Such a situation however also means that the data is only as safe as the users device is. When interacting with other instances all data transferred is encrypted via a secure connection. This leads to having personal data automatically protected when trading it with the validating instance.

While the data is protected when transferring it, it is also protected on the blockchain. Every request lifecycle is generating a new databox which is not linkable to the user or to any other databox. This means that it cannot be exposed and the databoxes cannot be reverted to reveal the unencrypted and unhased data stored.

Because the application on the blockchain is readable for everyone the purpose of the application is also communicated to everyone. However because not everyone can read source code the application in a live environment needs a introduction why and which data is collected. The prototype offers a web interface in which the steps are communicated to the user as well as the storage on the blockchain. Only data which is needed for the calculation is stored and processed.

Only the validating instance has the possibility to link the storage to the user. All other instances do not have such possibilities to know which user belongs to which databox. No personal data is available to other instances except the validating instance. The other instances just work with the provided databox on the blockchain.

Privacy Embedded into Design

The prototype enables to have privacy by design as a core functionality. This is possible because the user keeps all the data and can handle it within the enclave running on the device. Having an encrypted connection between each instance as an addition as well as storing only hashed and encrypted data enhances the privacy as core functionality. The user can decide to delete the data from the device at any time without needing another instance. No request to a third party company is needed and the user can be ensured that all data regarding the calculation on the own device is deleted.

Because of having a prototype which is not provided by a company there are also no stakeholders which need to be consulted. However having this application running in a live environment every key person of each instance needs to be consulted to avoid errors in the terms of privacy by design. This also means that every instance needs to know how to handle a potential error within the application itself.

Full Functionality – Positive-Sum, not Zero-Sum

The functionality of the calculation in this prototype is a simple summation of all E_itx to get the desired result as explained in Section 6.3.4. Because of this it is fairly easy to achieve. This means that the prototype has the same functionality as if it would be offered by a company. However when implementing this in a live environment it would still be possible to calculate everything within the enclave. The only potential limitation is the power of the used device which also needs the technological possibility to enable an enclave. Because such a calculation might be a very complex operation it might not be able to execute it on the blockchain.

This prototype has a very low complexity when examining the calculation itself. Having this situation in a live environment might be a problem due to the fact that the calculation itself might fail. This could happen if for example data is missing. In another application a company might provide a possibility to contact support. However since everything is handled on the users device, giving support might be a problem. The company cannot take a look at the data without revealing it. This is a trade off a user has to accept.

End-to-End Security – Lifecycle Protection

Data is transferred securely and encrypted from instance to instance with the shortest way possible. This is true for every connection. Also only data which is really needed is transferred from instance to instance. As an addition the system is designed in a way in which each request lifecycle is encapsulated within its own to ensure that data is not traded between requests of different lifecycles.

Being able to destroy data if it is not needed anymore is only partial true. When calculating the result on the blockchain the clear data used is instantly destroyed after the calculation is finished. The user is the only one in possession of the full data sets. When the calculation is happening on the users device nothing gets destroyed because it is on the users device anyways. However a problem with this might be that the hashes which are grounded on the blockchain cannot be destroyed completely due to

the persistency of the blockchain. Even if the hashes cannot be linked to users they are still part of the users data.

Visibility and Transparency

Having encrypted connections between instances and using the shortest way possible is an essential part of the prototype. Every piece of data is secured to avoid man in the middle or similar attacks. This also needs to be true when having the application in a live environment.

The calculation on the Blockchain is fully transparent because the source code is readable by everyone in this case. Even though it is offered users might not be able to read it due to the fact that they cannot read the source code. The application within the enclave however is not so easily readable. The software provider would need to publish the source code open source to enable some kind of transparency. In this perfect scenario of the prototype this is of course the case. Every piece of source code is available for everyone. In a live environment however it is depending on the software provider itself to publish the intellectual property. A workaround in the live environment would be the publication of a documentation.

For this prototype no active steps to monitor, evaluate and verify compliance are taken. This is on the one hand because all source code is publicly available and the rules can be read from there. On the other hand no other person tested the application for compliance or evaluated it. This must not be the case when having this application in a live environment. The application from the software provider as well as the whole process of course needs to be evaluated by a external compliance company. The company which is in charge of the evaluation needs to be trusted by the validation instance. Otherwise the validation instance might not accept results submitted by the proving instance.

Having redress and compliant mechanisms within a decentralized application is hard within such an architecture due to having a lot of different instances. Users can complain to the software provider or validating instance. This could be used when reporting bugs which need to be fixed or other errors. However the prototype application itself has no complaint or redress mechanism by default. In a live environment to company providing the software or the validating instance needs to have for example a contact email address for this.

Respect for User Privacy

Only individuals have access to their data per request life cycle. The individual can request the data from the data providing instances at any time as well as from the validating instance. This is of course only true for sensible data. Information which is stored on the blockchain can be seen at any time by anyone. However since the user knows which data box he has he can always see which data is stored in there and others cannot connect the data boxes of a specific user.

Because the data is always on the users device challenging the accuracy and completeness of the data is possible for the user at any time without any problems. The user has the data and can, if desired, even calculate the result manually. The source code is publicly available on the BLockchain which also means that the algorithm for the

Table 7.1: Evaluation of the calculation within prototype environment.

	Enclave	*Blockchain*
Proactive not Reactive	no	no
Privacy as the Default	yes	yes
Privacy Embedded into Design	yes	yes
Full Functionality	yes	yes
End-to-End Security – Lifecycle Protection	yes	yes
Visibility and Transparency	partial	partial
Respect for User Privacy	yes	yes

calculation is publicly available as well. If the data is not accurate, complete or leads to wrong results (from for example the data providing instances) a simple hash check on the blockchain can show the faulty data which is not in sync. If the check fails the user can simply generate a new request lifecycle to re-fetch all the data again and recalculate the result with the new data.

7.2.2 Principles Summary

Having evaluated each principles by themselves a summary of which calculation method and environment the principles are complying with is generated based on the detailed evaluation of each single principle. Having a summarized view is more important than having an analysis each by itself because the system is only compliant if all seven principles are valid. The summaries contain the principles with an explanation as well as a yes, partial and no property. Yes and no are indicators if the principle can be completely fulfilled and comply with the defined tasks or not. Partial means that the principle cannot fully be fulfilled with the setup used and may need some changes to have it completely fulfilled. The two Tables 7.1 and 7.2 are also discussed in detail why several principles are only partial possible, how changes could be made to fully comply with the principle or give and short explanation why the principle is completely possible.

Table 7.1 summarizes the evaluation within the prototypical environment for the calculation within the enclave and blockchain. Most of the principles can be argued that they can be fulfilled. However the principle Proactive not Reactive cannot be fulfilled because the prototype does not allow to update the blockchain application once it is running. Even though it is designed in a way to prevent privacy evasive attacks having the system without being able to correct errors once it is running is considered as no go. Other principles such as the Full functionality are fulfilled due to the fact, that the enclave is only simulated and working on any device with the full set of features.

The End-to-End Security and Life cycle Protection are considered as fulfilled even though the hashes cannot be destroyed which are grounded on the Blockchain. Even though the hashes are part of the users data they cannot be linked to the user itself and also not reverted to the original data. Because of this the grounded data is considered as anonymous and therefore not classified as critical user specific content.

Table 7.2: Evaluation of the calculation within live environment.

	Enclave	*Blockchain*
Proactive not Reactive	yes	yes
Privacy as the Default	yes	yes
Privacy Embedded into Design	yes	yes
Full Functionality	yes	partial
End-to-End Security – Lifecycle Protection	yes	yes
Visibility and Transparency	yes	yes
Respect for User Privacy	yes	yes

Even though the calculation is completely transparent and the source code is publicly available the principle of Visibility and Transparency cannot be considered as completely fulfilled. The steps to monitor, evaluate and verify are completely missing within the prototypical environment. Because of this only two of the four defined requirements can be considered as fulfilled. Therefore this principle is only partially fulfilled and may need changes to have it completely fulfilled.

The summarized evaluation of Table 7.2 is a consideration when having the designed architecture within a live environment. Compared to the prototypical environment the Proactive and Reactive principle is considered as completely fulfilled. The major reason of this is that within a live environment no Javascript based blockchain would be used. The application needs to be built on top of a Blockchain technology which allows the upgrade or adaption of the application or the possibility to forward requests to the required version of an application if multiple versions exists.

The principle of Privacy Embedded into Designed can be fully fulfilled if every key person of every instance is consulted. This of course means that the architecture itself does not fulfill this principle itself but it is in the hand of the company building the application and consulting the other instances. If the company does not offer consulting or a proper documentation the principle would fail. However since this application and situation is in a perfect environment it is assumed that every instance and person is consulted.

The full functionality when having the calculation on the blockchain might not be possible because it is too cost consuming if the calculations complexity is too big. The price of a function call depends on the complexity and size of the data transferred and calculated. If there are a lot of transactions or data needed for the calculation it might get too complex to be calculated. This is depending on the blockchain used. When having the calculation in the enclave no problems should occur if the user has a device which supports the enclave functionality. When taking a general look at the technical part it is very hard to put such an application with all features within a live environment. Theoretically if every instance is working together it is working.

7.3 Issues and Conflicts

The prototype is simulated in a perfect environment but even in this environment the architecture cannot fulfill all principles of Privacy by Design. However the evaluation displayed also potential problems to be able to comply with Privacy by Design when having the prototype in another environment.

A major issue when planning to have the architecture in a live environment is the initial setup which cannot be changed easily. When choosing a blockchain for the live environment it of course needs to be a public blockchain to be transparent for every instance and user using the application. The blockchain monitor as introduced in Section 6.4.3 needs to be adapted depending on the blockchain to have the full transparency given at any time. Having multiple blockchains in the architecture might allow an easier swap between the blockchains if for example a blockchain stops existing but it also increases the complexity when trying to update the application.

Having a general look on the definition of the principles of privacy by design it gets clear that when trying to fulfill all principles it is heavily depending on having a company behind the software. Even if the principles itself are defined as guidance, it is not defined how for example open source software needs to be handled. Not to mention how a decentralized application should be handled. This leads to a lot of room for interpretation which could lead to disputes when interpreting it different than another person.

Having an evaluation of Privacy by Design actually depends heavily on the final implementation. It is possible to evaluate the perfect scenario within this prototype but may get drastic complex when having multiple companies in the environment. Each company might have their own processes and might not be able to adapt those. The architecture however can only be considered as fulfilled if every instance allows the adaptation. Having this in mind there might be the need to research on how to loose coupling the instances to decrease their responsibility.

The use of the used hash generation and asymmetric encryption may not lead to problems for now but maybe in the future. As stated in Chapter 3 there are already mechanisms which attack the problems quantum computers might bring. Not only that the hashes generated in this prototype are theoretically being able to attacked with a dictionary attack due to the missing salt and randomness but also the encryption could lead to problems when being able to be decrypted. Even if this was not part of the prototype it needs to be mentioned. With this in mind the hashing should be done with a quantum safe hashing algorithm such as SHA3-256 as described by Amy et al. in [1].

All in all it can be said that having a Zero Knowledge system, which complies with the for this prototype created terms of Privacy by Design, is possible. The blockchain however might be a limitation due to the generated costs when executing functions with high complexity or much data. The blockchains are getting more and more mature and might allow an easier implementation in the future. When having the enclave there are fewer limitations for now. However issues occur in combination with processors which of course would also lead to problems with the enclave.

The latest discovery for example discovered by Schwarz et.al. in [15] allows to intrude processors and therefore also the enclave running on it:

> *While programs normally only see their own data, a malicious program can exploit the fill buffers to get hold of secrets currently processed by other running programs. These secrets can be user-level secrets, such as browser history, website content, user keys, and passwords, or system-level secrets, such as disk encryption keys.*

Having this in mind it can be said that the blockchain application is more manipulation safe but has the disadvantage due to the need of sending a lot of data through the network. This could also be avoided if the user has the full blockchain installed on the own device. This, however, may lead to problems with the storage due to the fact that the sizes of the blockchains are increasing consistently. For example as of May 2019 the size of the Ethereum Blockchain is already over 326 GB.

Chapter 8

Summary

The goal of the thesis was to design and implement an architecture focused on a Perfect Zero Knowledge system which meets the requirements of Zero Knowledge Proofs. The prototype shows an application in which a user can process data, which normally would be handled by a third party company, on their own device. With this approach the user does not have to give away his personal data but still let others the possibility to validate the result of the processed data. With the prototype developed an evaluation against the principles of privacy by design within a perfect test environment shows if it complies with the principles. As a second evaluation architecture was evaluated within a fictive live environment.

Even if Zero Knowledge applications, in which parts of the business logic is running on the blockchain, are definitely possible by now technical limitation occur if the business logic is too complex. A combination of running parts of the application within the enclave on the users device and having validations running on the blockchain might be the most effective solution for now. The limitations of the blockchain used of course vary from technology to technology. However since the technology of the blockchains get more mature every day there might be the possibility to run the whole application within the blockchain in the future. Letting the user install the blockchain as mentioned in Section 7.3 is way too much overhead and the only benefit the user has is to not having to send the data through the network. However an ideal application should let the user decide with which blockchain nodes the application should be connected. This could be a remote node but also a local node then. Such decisions must be observed before starting to build the application due to the high complexity of updates after the initial deployment.

Even if the logic of the prototype is kept fairly simple it shows how Zero Knowledge Validations with multiple data providing instance can theoretically work within a live environment. The prototype works on a single device and therefore in a perfect environment. Because of this the evaluation is not as complicated as evaluating the application in a live environment. In my opinion such an application will not work when having it in a real live environment with the given situation. Not because the technology is the limitation but the problem will be to convince every instance necessary to implement the solution.

Because every application which wants to have a Zero Knowledge proof validation is different it cannot be generally said that such an application complies with the principles of Privacy by Design or not. Even if this prototype complies with some principles it does not mean that another application with a Zero Knowledge proof validation complies with them as well. This of course means that another application can comply with more but also less principles. To have a proper validation for each application the principles need to be validated for each application independently. This evaluation needs to be communicated to the user to enable transparency here as well.

In my opinion the thesis shows a good example of the possibilities to avoid having centralized databases and give users back more control of their data while providing full functionality. When having a look at the principles of Privacy by Design it can be said that the description of the principles has some kind of room for interpretation. The principles should theoretically work for technologies, processes, companies and a lot of other fields but because every field has different challenges the principles should not be generalized. This could lead to problems in which company A interprets it one way while company B interprets it in another way. The principles should be seen as guideline, defined in detail for every use case and communicated to the user in an open way. It also needs to be defined for which principles the product or company draws a line. The deletion of the data hashes on the blockchain even though it cannot be reverted to the original data is a good example for this.

It is exiting to see how the technologies behind the blockchains matures, give more and more use-cases to work with and speculate where the future leads.

Appendix A

CD-ROM Contents

Format: CD-ROM, Single Layer

A.1 PDF-Files

Path: /pdf_files

thesis.pdf	Digital Version of this Master Thesis

A.2 Project-Files

Path: /project

blockchain	Folder containing the source code for the Blockchain
exchange	Folder containing the Source code for the data providing instances
proving_instance	Folder containing the Source code for the proving instance
validating_instance . .	Folder containing the Source code for the validating instance
start.sh	Script to start all instances at once

A.3 Online-Sources

Path: /online_sources

literature	Collection of Online Sources used in this Thesis

References

Literature

[1] Matthew Amy et al. “Estimating the cost of generic quantum pre-image attacks on SHA-2 and SHA-3”. In: *Selected Areas in Cryptography – SAC 2016*. (St. John’s, NL, Canada). Basel, Switzerland: Springer International Publishing, Oct. 2016, pp. 317–337 (cit. on p. 47).

[2] Nir Bitansky et al. “From Extractable Collision Resistance to Succinct Non-interactive Arguments of Knowledge, and Back Again”. In: *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*. (Cambridge, MA, USA). New York, NY, USA: ACM, Jan. 2012, pp. 326–349 (cit. on p. 12).

[3] Sean Bowe, Ariel Gabizon, and Matthew D. Green. “A multi-party protocol for constructing the public parameters of the Pinocchio zk-SNARK”. In: *Financial Cryptography and Data Security*. Berlin, Heidelberg: Springer, 2017, pp. 64–77 (cit. on pp. 14, 15).

[4] Ferdinand Brasser et al. “Software Grand Exposure: SGX Cache Attacks Are Practical”. In: *Proceedings of the 11th USENIX Conference on Offensive Technologies*. (Vancouver, BC, Canada). Berkeley, CA, USA: USENIX Association, Aug. 2017 (cit. on p. 16).

[5] Benedikt Bunz et al. “Bulletproofs: Short Proofs for Confidential Transactions and More”. In: *2018 IEEE Symposium on Security and Privacy (SP)*. (San Francisco, CA, USA). San Francisco, CA, USA: IEEE, May 2018, pp. 315–334 (cit. on p. 13).

[6] Ann Cavoukian. “Privacy by Design: Origins, meaning, and prospects for assuring privacy and trust in the information era”. In: *Privacy Protection Measures and Technologies in Business Organizations: Aspects and Standards*. Ed. by George O.M. Yee. IGI Global, 2012. Chap. 7, pp. 170–208 (cit. on pp. 2, 8, 10).

[7] Michael Crosby, Nachippan Pattanayak Pradhan, and Sanjeev Verma Kalyanaraman Vingesh. “BlockChain Technology, Beyond Bitcoin”. *Applied Innovation Review* Issue 2 (June 2016), pp. 6–20 (cit. on p. 4).

[8] Ben-Sasson Eli et al. “Scalable, transparent, and post-quantum secure computational integrity”. Unpublished. 2018. URL: https://eprint.iacr.org/2018/046.pdf (cit. on pp. 12, 13).

[9] *European Data Protection Supervisor Opinion 5/2018 Preliminary Opinion on privacy by design*. 24941/EU XXVI.GP. May 2018. URL: https://edps.europa.eu/sites/edp/files/publication/18-05-31_preliminary_opinion_on_privacy_by_design_en_0.pdf (cit. on p. 8).

[10] Uriel Feige, Amos Fiat, and Adi Shamir. "Zero-knowledge proofs of identity". *Journal of Cryptology* 1.2 (June 1988), pp. 77–94 (cit. on p. 5).

[11] L. Fortnow. "The Complexity of Perfect Zero-knowledge". In: *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*. (New York, NY, USA). Ed. by Alfred V. Aho. New York, NY, USA: ACM, 1987, pp. 204–209 (cit. on p. 7).

[12] Oded Goldreich. *Zero-Knowledge twenty years after its invention*. Tech. rep. Electronic Colloquium on Computational Complexity, 2002 (cit. on p. 5).

[13] S. Goldwasser, S. Micali, and C. Rackoff. "The Knowledge Complexity of Interactive Proof-systems". In: *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*. (Providence, Rhode Island, USA). New York, NY, USA: ACM, May 1985, pp. 291–304 (cit. on pp. 1, 7).

[14] Ira Rubinstein and Nathan Good. "Privacy by Design: A Counterfactual Analysis of Google and Facebook Privacy Incidents". *Berkley Technology Law Journal* 28.2 (2012) (cit. on p. 8).

[15] Michael Schwarz et al. "ZombieLoad: Cross-Privilege-Boundary Data Sampling". 2019. URL: https://arxiv.org/abs/1905.05726. Pre-published (cit. on p. 48).

[16] Melanie Swan. *Blockchain: Blueprint for a New Economy*. 1st ed. Wiesbaden, Germany: Springer Fachmedien, 2015 (cit. on p. 5).

[17] Salil Pravin Vadhan. "A Study of Statistical Zero-knowledge Proofs". PhD thesis. Cambridge, MA, USA: Massachusetts Institute of Technology, 1999 (cit. on p. 6).

Online sources

[18] Binance Academy. *zk-SNARKs and zk-STARKs Explained*. Feb. 2019. URL: https://www.binance.vision/blockchain/zk-snarks-and-zk-starks-explained (visited on 04/08/2019) (cit. on pp. 12, 13).

[19] Beanstalk. *Demystifying Zero-Knowledge Proofs*. Apr. 2018. URL: https://docs.google.com/presentation/d/1gfB6WZMvM9mmDKofFiblgsyYShdf0RV_Y8TLz3k1Ls0/edit#slide=id.p (visited on 04/08/2019) (cit. on p. 13).

[20] Daniel Benarroch. *Trustless Computing on Private Data*. June 2017. URL: https://qed-it.com/2017/06/08/trustless-computing/ (visited on 04/08/2019) (cit. on p. 15).

[21] Rostyslav Demush. *How Companies Can Leverage Private Blockchains to Improve Efficiency and Streamline Business Processes*. URL: https://perfectial.com/blog/leveraging-private-blockchains/ (visited on 04/07/2019) (cit. on p. 5).

[22] Gemalto. *Analysis: Data breaches compromised 4.5bn records in half year 2018*. Oct. 2018. URL: https://thecitizenng.com/analysis-data-breaches-compromised-4-5bn-records-in-half-year-2018-gemalto/ (visited on 04/04/2019) (cit. on p. 2).

[23] Martin Hron. *Top 10 Biggest Data Breaches in 2018*. Dec. 2018. URL: https://blog.avast.com/biggest-data-breaches (visited on 04/04/2019) (cit. on p. 2).

[24] Preethi Kasireddy. *ELI5: What do we mean by "blockchains are trustless"?* Feb. 2018. URL: https://medium.com/%5C@preethikasireddy/eli5-what-do-we-mean-by-blockchains-are-trustless-aa420635d5f6 (visited on 04/07/2019) (cit. on p. 7).

[25] Hanna Kozlowska. *The Cambridge Analytica scandal affected nearly 40 million more people than we thought*. Apr. 2018. URL: https://qz.com/1245049/the-cambridge-analytica-scandal-affected-87-million-people-facebook-says/ (visited on 04/04/2019) (cit. on p. 2).

[26] Lucas Nuzzi. *Monero Becomes Bulletproof*. Oct. 2018. URL: https://medium.com/digitalassetresearch/monero-becomes-bulletproof-f98c6408babf (visited on 06/08/2019) (cit. on pp. 16, 17).

[27] Tom Robinson. *Bitcoin is not anonymous*. Apr. 2015. URL: https://www.republica.org.uk/disraeli-room-post/2015/03/24/bitcoin-is-not-anonymous/ (visited on 04/07/2019) (cit. on p. 11).

[28] Saurav. *A Primer on Zero-Knowledge Proofs*. Jan. 2019. URL: https://hackernoon.com/a-primer-on-zero-knowledge-proofs-892e6e277142 (visited on 04/07/2019) (cit. on p. 7).

[29] Lukas Schor. *On Zero-Knowledge Proofs in Blockchains*. Mar. 2018. URL: https://medium.com/@argongroup/on-zero-knowledge-proofs-in-blockchains-14c48cfd1dd1 (visited on 04/07/2019) (cit. on p. 6).

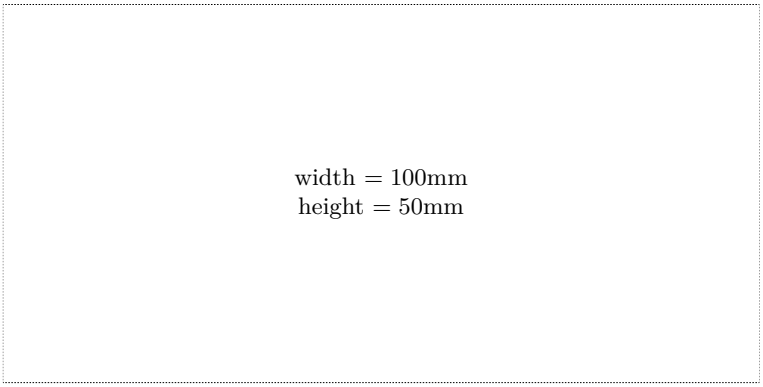
[30] Z-Cash Team. *What are zk-SNARKs?* URL: https://z.cash/technology/zksnarks/ (visited on 04/07/2019) (cit. on pp. 12, 14, 17).

[31] David Terr. *Polynomial Time*. URL: http://mathworld.wolfram.com/PolynomialTime.html (visited on 06/17/2019) (cit. on p. 6).

[32] Reuben Yap. *Zcoin moving beyond trusted setup in Zerocoin*. Apr. 2017. URL: https://zcoin.io/zcoin-moving-beyond-trusted-setup-in-zerocoin/ (visited on 04/07/2019) (cit. on p. 12).

Check Final Print Size

— Check final print size! —



— Remove this page after printing! —