VRIJE UNIVERSITEIT AMSTERDAM

A Distributed Computation Engine for Reproducible, Verifiable, Reusable and Transparent Computations

by

Alex Oberhauser

A thesis submitted in partial fulfillment for the
degree of Master of Science

in the
Faculty of Science
Department of Computer Science

January 2016

Declaration of Authorship

I, Alex Oberhauser, declare that this thesis titled, 'A Distributed Computation Engine for Reproducible, Verifiable, Reusable and Transparent Computations' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a master degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"To follow the path, look to the master, follow the master, walk with the master, see through the master, become the master."

Zen Proverb

VRIJE UNIVERSITEIT AMSTERDAM

Abstract

Faculty of Science
Department of Computer Science

Master of Science

by Alex Oberhauser

Reproducibility and verifiability are core scientific principles, but nevertheless they are not easily achievable for scientific computations. The main reason is the immense computational power and storage needed for some of the computations. Another reason is the storage of the results in data silos that are only limited accessible. This work proposes a solution to the problem by introducing a distributed blockchain or public ledger, that keeps track of all computations and related data, makes them verifiable, reusable and reproducible. The resulting overlay network makes scientific computations not only more transparent, but also enables sharing of storage and computation resources. Additional to the specification of the blockchain and the related protocol, the work provides a proof of concept implementation.

" If I have seen further it is by standing on the shoulders of giants."

Isacc Newton

Acknowledgements

I want to express my gratitude to my amazing supervisor team, Dr. Adam Belloum, Dr. Ana Oprescu and Dr. Reggie Cushing, which not only supported me throughout the thesis, but also made it happen that the thesis resulted in multiple research proposals.

Furthermore, I want to thank my two co-founders Corneliu Stanciu, for early discussions and support through the early phase of the thesis and Matthew Commons for founding with me a new company, based on the findings of this thesis. Also a big thanks to the whole blockchain space and the countless people that crossed my path during this time and helped me to form the ideas outlined in this thesis.

I also want to thank my blockchain team in Amsterdam for the parallel work on a complementary project: Stavros Choampilomatis, Nikos Sarris, Maria Efthimiadou and Ilias Diamantakos.

Last but not least, I want to thank my parents and my brother, Stefan, for supporting me throughout the time and making it even possible that I could pursue my education and more importantly my dreams.

Thank you all!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"Science is nothing but perception."

— Plato

If science is nothing but perception, access to scientific results, as well as all intermediate and related knowledge, is essential to form a grounded and scientific opinion about a topic or field and build on top of it. That was trivial in the past centuries, because one person could learn the whole knowledge of humanity, although it is impossible in the 21^{th} century. Take for example early scientists, they had a holistic understanding of all human scientific knowledge. That is only possible if the available knowledge is comprehensive and not too extensive. In comparison the modern scientist is highly specialized and not able to have in-depth knowledge about his or her own field, only in a specific subfield. Human brains are evolutionary hardwired for survival and not for data and computational intensive scientific work [Cha13]. As compensation we rely on technology, but in the end a human scientist has to evaluate and use the results.

1.1 The Problem

Technology so complex that it could not be understood by one single person, introduces a set of new problems. For example the internet, originally meant as free and open platform, accessible by everybody and as *"knowledge database"*, has moved towards islands of data silos. This trend of vendor lock-in, could be seen in social media platforms (Figure 1.1), in mobile app stores (e.g. Apple App Store, Google Play), for digital contents (e.g. Amazon Kindle), but also in scientific communities. This paper will focus on the latter and propose a potential solution, that could be deployed on top of the current infrastructure, without violating existing workflows used by scientists. Vendor

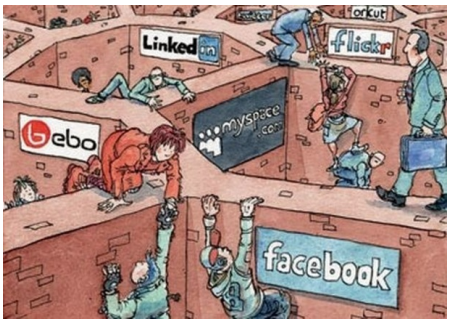


FIGURE 1.1: Depiction of social media data silos. The same trend could also be seen in other parts, e.g. for digital contents, like ebooks, or app store, but most importantly and worrisome for scientific results.
Source: Unknown/Various, has gone viral a few years ago.

locking may have a place in industry, but, except for well defined border cases, not in science. One example of an exception is medical research, involving confidential patient data.

In order to pinpoint the concrete problem - why most scientific results are only reproducible, verifiable and reusable to some extend or not at all - the problem is broken down into the following high-level categories:

Technology Heterogeneity: The vast amount of different computation and data storage frameworks, but also the existence of closed and private infrastructure, creates data silos, with inaccessible scientific results. Also if results are published, valuable information is lost, e.g. the path to the results via traceable intermediate steps.

Result and Data Publication Process: The published results are only part of the complete experiment or computation. Moreover interpretations of raw data can have a subjective bias, e.g. by cherry-picking suitable data [Bal15]. Without access to the original data and without a method to verify the results by reproducing them, it is nearly impossible to verify the overall scientific claim.

Publisher Access Rights: In most cases the publisher gets all access rights to the publication and the resulting commercialization, by making the papers only accessible via expensive subscription fees, makes them even less accessible.

This paper tackles the challenge of **Technology Heterogeneity** and provides in the process a solution to the **Result and Data Publication Process** issue. The **Publisher Access Rights** point is a social-economical problem and can not be solved with a scientific or technical solution. Hence, it will be ignored in the scope of this work, although the author encourages the scientific community to face this well known and

FIGURE 1.2: *The Economist*, issue October 31st - November 6th 2015, featuring Blockchain technology

behind-closed-door discussed issue and start a public discussion, with the goal to separate commercial from scientific interests.

1.2 The Enabler Technology

The emerging of new technologies allows the evaluation of previous challenges and the investigation of new potential solutions. The same way as the internet has created a new ecosystem and changed society and the economy in a fundamental way, a new big technology wave may be on the rise. This new technology, with the name *"blockchain"*, was first introduced as the digital currency Bitcoin [Nak08], at the end of 2008, and has been applied since then in a variety of new applications. The fact that this technology solves the double-spending problem without introducing a centrally controlled clearing house, makes it suitable for a trust-less distributed ledger for non-copyable assets. The encapsulating of transactions into blocks, chaining of these blocks and distribution to all involved nodes makes historical transactions tamper-resistance. Each node cryptographically verifies the validity of this chain and all involved parts[1]. In contrast to classical distributed databases, there is no trust-relationship between the nodes and the historical data can not be changed anymore.

Most of the involved pieces of a blockchain, e.g. the underlying cryptography, are not new, but the smart combination of them has triggered a discussion. This discussion goes

[1] Transactions and Blocks

far beyond the technology and has reached, since the first emerging of the digital currency Bitcoin in 2008, mass media and even conservative sectors, like financial institutions[2] and was featured on the cover of *The Economist* (November 2015, see Figure 1.2). Additionally, non-financial alternatives are investigated by different parties. A good non-technical overview about blockchains is presented in the book *Blockchain - Blueprint for a new economy* by Melanie Swan [Swa15a]. As of the writing of this paper, most parts of the book are accurate, but that could change pretty quickly. As expected from new technology driven fields, progress is measured in weeks, not in years or even decades. A more futuristic paper was published by the same author under the title *Blockchain Thinking: The Brain as a DAC (Decentralized Autonomous Organization)* [Swa15b]. The vast amount of projects and startups emerging in the space and the discussion of futuristic, even crazy, ideas, but also the joining of more serious players, gives a good indication that the technology is here to stay.

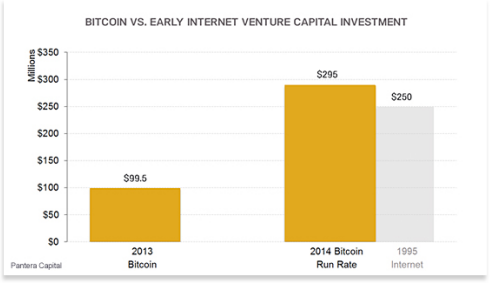


FIGURE 1.3: Comparison of VC investment in Bitcoin and the early internet.
Source: Pantera Capital - https://panteracapital.com/.

Blockchain technology is often compared to the internet in 1995, only time will show if that statement is an over- or underestimate, but the venture capital investments as shown in Figure 1.3 seam to support that claim. Also if the numbers are optimistically chosen in favor of Bitcoin and inflation is not considered, it shows investments in Bitcoin and not in the broader blockchain technology. Moreover, the general spirit of optimism and the reaction of different stakeholders to this new technology is comparable to the early internet age. A source, that underlies this point, is the blog post *Why Bitcoin is and isn't like the internet* by Joichi Ito (Director of MIT Media Lab) [Ito15].[3] Indications of the importance of this technology are the number of related meetups and the involvement major financial institutions. Like the internet before, the impact seams to be omnipresent and spans across social classes and nearly all verticals.

[2] For example *Banks and exchanges turn to blockchain* published in the *Financial Times*
[3] As of this writing Bitcoin core developers have joined the MIT digital currency initivate.

1.3 Proposed Solution

In this thesis the above mentioned technology is used to propose and develop a distributed data structure, with related protocol, with minimal disruption and intrusion to current workflows and frameworks. Instead of proposing yet another framework to solve the problem, a technology agnostic solution, that could be integrated into the existing landscape, is developed. After analysing the problem and investigating past, unsuccessful solutions the following properties were extracted:

Technology Agnostic There is no need to reproduce current, available and well functional infrastructure, e.g. computation and workflow engines. It is far more effective to develop a plug-and-play solution that could be integrated into different technology stacks; from home computers and interent of things, via cloud services, to grid infrastructure.

Transparent Scientific results must be traceable in a transparent way. Each involved and interested party must be able to verify the validity of past computations.

Non-Intrusive The proposed solution must fit into current scientific workflows, without the need to re-educate researchers.

Trusted The overall network and the confirmed results must be trustworthy, also if singles nodes are not. More important than preventing malicious nodes from submitting wrong results, is the cross validation of scientific results by multiple nodes or entities.

With these key properties in mind, it is getting clear, that instead of another computation and workflow engine, a trusted and cross-organizational meta-data store, on top of existing infrastructure, is needed. This meta-data store should keep track of computation requests and performed computations.

In this chapter, a short introduction to the problem was given. Additionally, the used technology was introduced and the potential impact of it was explained. Furthermore a first path to a solution was shown. Continuing with chapter 2, past and current approaches are outlined and analysed. That includes not only related work about reproducible, verifiable, reusable and transparent computations, but also related work about the key technical concepts, used to solve the higher level problem. Chapter 3 bridges the gap between the introduced blockchain technology and scientific computations. It furthermore explains why this technology is key to solve this long standing problem. Before describing in detail the solution and specifying all involved parts (chapter 5), a

multi blockchain approach with different extensions is introduced in chapter 4. This unconventional order should help the reader to understand the importance of sharable transactions and what side-effect such a system has. Without limiting authority of different parties, it can be nevertheless assured that computations can be shared and reused. The accompanying implementation is described in chapter 6. In the following chapter 7 the results are discussed. The paper concludes with chapter 8.

A blueprint how to develop a new blockchain can be found in Appendix A, followed by the binary specification of transactions in Appendix B and blocks in C.

Chapter 2

Related Work

In this chapter the related work is outlined. It contains a domain-specific section, that explains past and current efforts to make scientific computation reproducible. Additionally, it contains a technology section that describes available solutions which can be used to solve the domain-specific problem directly or indirectly, by combining multiple approaches.

2.1 Reproducible Scientific Computations

Making scientific computations reproducible is not a new problem and researchers across fields agree that science must be more collaborative. The raise of Big Data has not solved the problem, but made the collbaration between groups even more complex, e.g. collaboration with external parties, not originally involved in the research project. Different fields have tried to tackle that problem in variable ways. For example earthquake engineers have used a more social approach for their own field [FJ10], by trying to establish standards for collaboration. In comparison other parties have tried to solve it from a technical perspective [Jha+07; KKS08; Bec+13; SKC00]. A closely related paper to the presented approach is *Toward Executable Scientifc Publications* [Str+11]. The here presented approach does not directly specify a workflow and does not embed a workflow into publications, but specifies a URI schema that can be used to reference a single computation, as well as track and verify all related (linked) computations from there. In difference to domain-specific solution of earthquake engineers, this paper is domain-agnostic and can hence be used across all fields. The past technical solutions are based on one or more specific frameworks or technology stacks, e.g. Grid infrastructure. This

tightly coupling with existing frameworks introduces problems in perspective of obsolescence. A more sustainable and future proof approach is to separate the computation and storage frameworks from the part that makes the scientific computations reproducible.

2.2 Consensus at Large Scale in a Trust-less Environment

The most challenging part of a trust-less, large scale distributed ledger is to reach consensus via node majority. In order to guarantee consensus related to executed computations and to verify, via majority votes, the correctness of the results, the network has to agree on valid transactions and encapsulate only them into blocks. Based on the fact that each block can only have one predecessor and one or none (if currently last) successor, the network has to agree on both the block order and content. Most importantly, each node must have the same chain of blocks with the same ordered list of transactions per block. More details and the concrete structure are describe later in the paper.

2.2.1 Pre-Blockchain Consensus Algorithm

The consensus problem in an untrusted environment was first described in the paper *The Byzantine Generals Problem* [LSP82], published in 1982, and is now known under the same name. Over the years different extensions and modifications were proposed [C+99; Rei85; Bra87] in order to overcome the scalability problem and the static nature of the algorithm (all nodes have to be known upfront). For more than two decades, researchers in the field of computer science have tried to solve the Byzantine Generals Problem in a more generic way, without completely solving it. The major challenges, till today, are scalability and the dynamic nature of peer-to-peer systems, nodes join and leave over time. Algorithm based on the original proposed solution to the Byzantine Generals Problem require a fixed set of nodes, e.g. $2f+3$ nodes to guarantee correctness if less then f nodes are malicious or failing, or a slight variation of it. Moreover the need to exchange a multitude of messages between nodes hinders scalability. Another problem is the existence of a *broadcaster* that propagates a value on which all correct processes agree. It is shown later in the paper that in the here presented case there is no single *broadcaster* and theoretically all nodes can broadcast potentially a different value. A value in this case is a new block with an uniquely ordered transaction list, that was not part of any previous block and is valid.

There are more consensus algorithms for closed and small scale networks, like Raft [OO14], a simplified version of Paxos [Lam05], but with the same efficiency and shortcomings.

Why a Blockchain overlay network has no single broadcaster?

In the context of the first blockchain designed and implemented, Bitcoin, there are multiple broadcasters aka. multiple nodes that are generating or mining blocks. Conflicting blocks are minimized by making it computational expensive to generate new blocks. The computational complexity is adapted by the network, based on the overall computation power availability, assuring an average block generation every few minutes. The incentive of a node to invest a vast amount of computation power is the gain of an asset that has value, if successful, e.g. Bitcoin.

Byzantine Generals Problem Description: As mentioned previously, the Byzantine General Problem was introduced the first time in 1982 [LSP82] and copes with failure of involved nodes under the assumption that the number of such entities is less then $\frac{1}{2}$. That means that $2 * f + 1$ nodes have to be deployed in order to assure that the majority of nodes reach consensus if a maximum of f nodes are failing. The communication overhead is high and scalability is limited, because each node has to communicate with each other. Also after decades of research and the proposal of different modifications these problems couldn't be solved completely.

In summary that means the originally proposed solutions to the Byzantine Generals Problem, including extensions and modifications, are not applicable for the following reasons.

Dynamic Nodes In the here proposed solution, nodes join and leave the peer-to-peer overlay network. This behaviour is not regarded as a fault, but as build-in functionality.

Internet Scale Depending on the popularity of the scientific computations the underlying data structure and the overlay network must be able to scale.

Multiple Broadcasters In comparison to the original approach, each node is able to broadcast a new value. This results in conflicting values, leading potentially to network forks and inconsistencies in the long run. A single broadcaster in this case would defeat the purpose of the solution and re-introduce a previously eliminated party.

2.2.2 Blockchain

As mentioned previously, Bitcoin [Nak08] was and is a game changer. Before the introduction of blockchain technology, more specifically the mining process and a consensus algorithm that prevents double spending of assets, digital currencies needed a central

clearing house to check for double-spending. In hindsight that was the reason why previous digital currency approaches, like [MN93; PHS98; Cla98], failed. Sometimes Bitcoin is praised as solution to the Byzantine Generals Problem, although it is only applicable for a specific usecase and it does come with a price. In order to guarantee the functionality of the network and provide tangible value, the block generation (known as mining[1]) changes automatically the hashing complexity, based on the current available computation power. For example if the complexity is not adapted, but the computational power increases, more blocks are produced in a specific time frame. This behaviour assures that blocks are only generated after a couple of minutes (currently 10), minimizing the amount of conflicting blocks in the network. There are multiple downsides to that approach, one of which is the "wasted" computation power. One article from 2013 estimates the computation power of the whole Bitcoin network as 256 times faster than the top 500 supercomputers combined[2]. The article is most likely outdated and the invested computation is now even higher. A resulting problem is the emerging of ASIC miners. As of this writing the majority of mining is done with these Application Specific Integrated Circuit miners, that are designed for Bitcoin mining or any other hashing with the same algorithm. From a scalability point of view, the Bitcoin network can handle at the moment a maximum of 7 transactions per seconds, because of the block size restriction. In comparison, only the VISA network can handle around 10000 transaction per second at peak times[3].

The limitations of Bitcoin mining, called Proof of Work, or for short PoW, has motivated different parties to develop alternatives like Proof of Stake (PoS). In PoS nodes vote and are selected based on the percentage of how much they own of the whole network. Such alternative consensus approaches introduce a new set of problems and are not yet proven in practise. There are further approaches that are hybrid solutions or build on top of more classical consensus algorithm, presented in the previous section.

2.2.2.1 Proof of Work (PoW)

Used by: Bitcoin and most Altcoins[4]
Description: Consensus algorithm based on solving a complex mathematical problem, which needs a lot of computation power, but can be verified easily. The network automatically adapts the complexity of the problem, in order to guarantee that the network

[1] It is called mining, because the entity that generates the next valid block gets an amount of Bitcoin as compensation.
[2] Global Bitcoin Computing Power Now 256 Times Faster Than Top 500 Supercomputers, Combined! - http://onforb.es/1dcxVIN Accessed: 5. July 2015
[3] Bitcoin needs to scale by a factor of 1000 to compete with Visa. Here's how to do it. - http://wapo.st/1KuETz3 Accessed: 5. July 2015
[4] `http://coinmarketcap.com/` - accessed: 5. July 2015

is not flooded with conflicting blocks. In Bitcoin the network tries to assure that a block is generate approximately every 10 minutes. This approach solves the consensus by making it harder to generate new blocks and minimizes potential conflicts. Also if not 100% accurate[5], Bitcoin has shown that this approach is suitable for day-to-day use and furthermore can be used to build a distributed system that is capable to transfer and create a financial asset and prevent double spending.

Advantage: Deployed since the beginning of 2009 and has real financial value. The protected digital asset, Bitcoin, can be exchanged to fiat currency, like Euro or Dollar.

Disadvantage: Rather high economical costs to "only" prevent double spending, high latency (more than 10 minutes for first confirmation, up to an hour to be certain) and poor scalability (max. of 7 transactions per second). Computations can be used to solve scientific problems, e.g. cure cancer, but these altcoins are not yet widespread and if that problem is solved and the mining algorithm is not generic enough, the block generation algorithm falls back to the before mentioned problem of Bitcoin. Also not suitable for smaller system, because it is easier to get 51% majority and control hence the network.

2.2.2.2 Proof of Stake (PoS)

Used by: Peercoin[6], Tendermint[7] (membership based on proof of stake)

Description: Voting mechanism based on the percentage of owned assets of the whole network, posted collateral. Nodes with higher stakes are preferred. Different variation of the approach, e.g. stakes are all issued upfront or over time, deflationary stakes to increase fairness or penalties for bad behaviour.

Advantage: More economical, because costs are lower, also faster block generation, that could solve the scalability limitations of PoW.

Disadvantage: Not yet proven to work in a real world setting. Debatable if distributed consensus could be achieved with Proof of Stake [Poe15]. Additionally, "nothing at stake" problem, where a node can vote for both variants of a chain fork, because it has stakes in both branches, making double spending possible.

2.2.2.3 Stellar Consensus - SCP Quorum Slicing

Used By: Stellar[8]

Description: In the original paper [Maz] the algorithm is described as federated Byzantine agreement (FBA) with the Stellar Consensus Protocol (SCP). Instead of having a

[5] Conflicting blocks and chain can occur, but the majority of the network will then favour one. Additionally, the block generation of 10 minutes may vary.

[6] http://peercoin.net/ - accessed: 5. July 2015

[7] http://tendermint.com/ - accessed: 5. July 2015

[8] https://www.stellar.org/ - accessed: 5. July 2015

single, global consenus the approach introduces quorum slices (subset of nodes that agree on facts) that together determine a global agreement. In comparison to the other solutions the algorithm claims to satisfy all of the following properties: decentralized control, low latency, flexible trust, asymptotic security.
Advantage: Claims to be faster than Proof of Work with flexible trust and asymptotic security, based on cryptographic families with fine tunable parameters.
Disadvantage: Same as Proof of Stake, not yet proven to work in real world setting. Dependent on the setup of quorum slices by the user.

2.2.2.4 Other Approaches

Currently multiple entities experiment with different alternatives to the here outlined consensus algorithms. Most of the approaches have not yet reached a degree of maturity that allows to analyse them.

The only tested consensus algorithm at scale[9], with enough real value that justifies a large scale attack, is Bitcoin with Proof of Work. Proposed alternatives, like Proof of Stake and hybrid solutions are currently explored and deployed in smaller scale or not at all. That is also the reason why potentially each blockchain, in the here proposed blockchain ecosystem, can implement a different consensus algorithm. Additional to the advantage of choosing the right algorithm for the task at hand, it allows also to test and integrate new approaches, without breaking the ecosystem.

[9]Here scale means globally deployed and used as accepted payment method. The hard limit of 7 transactions per second is temporary ignored and will potentially be fixed in future releases of the protocol.

Chapter 3

Scientific Computations and the Blockchain

The previous chapter has described related work in the field of scientific computations and Blockchain. This chapter will bridge the gap between the two fields and describe why the Blockchain can make scientific computations, and computations in general, reproducible, but also verifiable, reusable and transparent.

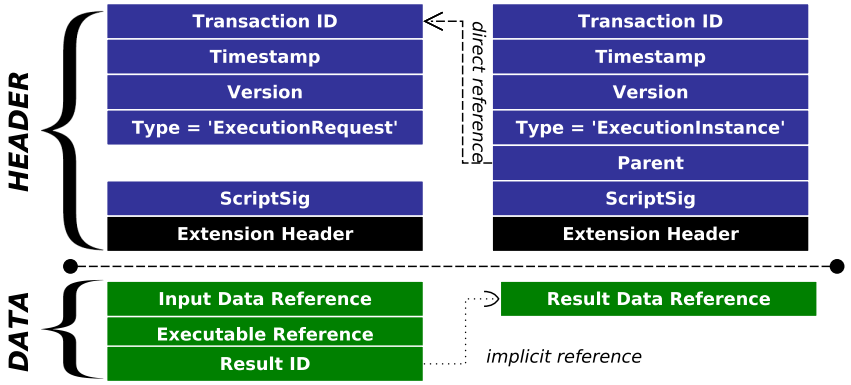


FIGURE 3.1: Visualization of the two, domain-specific, transaction types.

The here presented solution does not reproduce the functionality of computation frameworks, but extracts instead meta-data of an execution and stores it in an tamper-resistant and distributed way. Each execution request, or execution instance, is modelled in a transaction that is part of one blockchain. The main fields of an execution request are links to the input data (*Input Data Reference*) and the executable (*Executable Reference*). The related execution instances contain a link to the computed results (*Result Data Reference*), as shown in Figure 3.1. The technical details are given in the subsequent chapters.

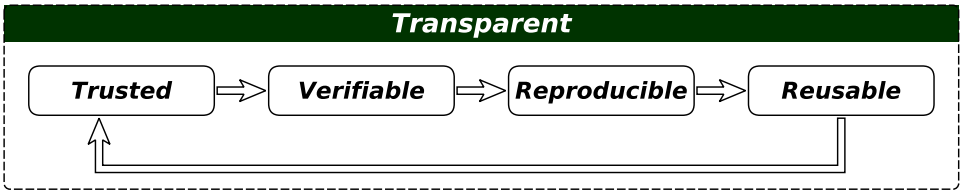


FIGURE 3.2: Chain of properties that result in a more transparent computation environment. If a transaction can be verified and comes from trusted sources the encoded computation can be reproduced and after that reused. That makes the whole environment more transparent and allows to trace results back to the origin.

Figure 3.2 visualizes the properties needed for a transparent environment. Computations are only executed if they were created by a trusted party. The linking of computations to any form of identity allows to make them cryptographically verifiable, resulting in a trusted and verified transaction, with related data, for execution or results, that can be safely reproduced and later reused to build on top of it.

3.1 Key Properties

3.1.1 Trusted

Trust is a delicate and subjective metric that is hard to quantify. The approach, as used in Blockchains, to mistrust single nodes, but trust the collective is also used for the domain-specific Blockchain ecosystem that is developed in the scope of this work. It basically means that computation requests are executed by multiple nodes. This redundancy comes with performance costs, but if executed on enough and independent parties, it could be assumed that the result as computed by the majority is correct. A malicious actor or group has to control more than half of the involved nodes.

If a number of malicious nodes mn is expected, the following formula computes the needed redundancy r to assure that the majority reaches the correct result. r nodes have to execute the same computation and publish the results to the network.

$$r = 2 * mn + 1$$

The formula calculates r, the number of nodes that have to execute the same computation request and publish the results to the network, if it is assumed that there are mn malicious nodes in the network. Then all, assumed correct, results of the majority are added to a new block and hence confirmed.

A high redundancy value decreases the changes that a malicious group could manipulate results, but comes with a performance penalty. It means that the overall network can execute less computation requests, because precious resources are used to prevent malicious intent. This approach assumes that the overlay network responsible for a specific Blockchain will have a certain degree of malicious players. In a completely untrusted network the value of expected malicious nodes mn will be higher than in a trusted environment. Also if assumed that all involved parties are trustworthy, e.g. if a Blockchain is used for a specific research project, were each involved party is part of a consortium, a certain degree of uncertainty should be expected. Also without malicious indent results could be wrong, e.g. because of error or malfunctions.

How to determine the number of redundancy? It can not be assumed that enough information is available to know the number of expected malicious actors in the network. Additional, if known this number will most likely change over time. The solution could be a build-in mechanism that determines this value automatically and constantly. By having, so called, reference computations with known results, malicious or malfunctioning actors can be detected. By combining that approach with a reputation system these actors can be banned or ignored, e.g. each wrong result that is propagated to the network decreased the reputation value and each correct result, that is included in a block and hence confirmed, increases it. One suitable algorithm could be the EigenTrust algorithm [KSG03], originally used for reputation management in peer-to-peer filesharing networks, or any other reputation algorithm for peer-to-peer networks. By combining the reputation mechanism with strong digital identities[1] Sybil attacks could be prevented, identities could not be easily forged.

3.1.2 Verifiable

Each computation request, with related execution instances, must be independently verifiable. An easy understandable and computational cheap verification process, based on cryptography, allows each involved or interested party to verify if the data source is really the entity that it claims to be (signed transactions) and that the published data was not altered (data hashes as part of the transaction). The process of detecting corrupted or intentionally modified data is shown in Figure 3.3. By using a multilayer protection approach, it is possible to detect changes of referenced and external stored data and transaction modifications. Referenced data is protected by simple file hashing and including these hashes inside a transaction. To assure that the transaction could

[1] Strong digital identities are governement issued IDs that are backed by cryptography and could be used in the digital world. One prominent example is the Estonia e-Residency - https://e-estonia.com/e-residents

not be modified by an unauthorized party, the issuer signs the transaction with a private key. It not only protects the transaction from changes, but proofs also the identity of the issuing party, increasing trust.

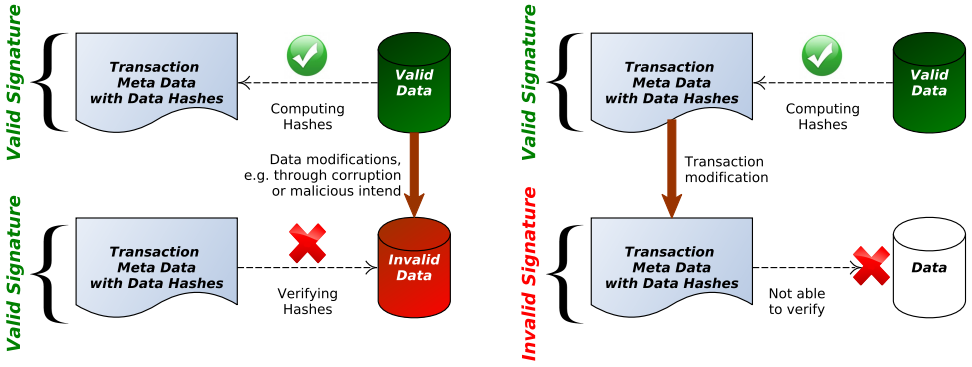


FIGURE 3.3: Detection of data or transaction corruption by multi layer protection.

As described in the previous section 3.1.1 only correct transactions can be part of a block. By using a majority vote and the right selection of redundancy the verifying node can be quite certain that results are correct. Moreover the chaining of blocks makes it over time harder and harder to modify past transactions. Combined with the fact that each node has the whole blockchain stored locally, makes it nearly impossible to compromise a once established fact about the validity of a transaction. In order to verify past computations, the node can access the local stored blockchain and verify all computations that are part of this blockchain. A valid and well behaving node will reject invalid transactions and blocks, making it even harder to inject invalid data.

3.1.3 Traceable

Trust (section 3.1.1) and verification (section 3.1.2) establishes an environment where nodes in the network can agree on basic facts, e.g. the correctness of computations, but that is not enough for scientific computations. Given a specific (final) result it must be possible to trace the result back to the origin and if necessary reproduce or reuse each intermediate step. Similar to workflow engines, where a specific description language is used, to define worklows, the here presented approach contains a build-in mechanism to define a flow of transaction by references. For starters there is always a reference from an execution instance, that publishes the computed results, back to the execution request, including the executable and input data. Additional to the possibility to have different data source links, e.g. a p2p filesharing magnet link or data on a shared cloud storage, data can be referenced from a previous transaction. This makes it possible to use the result of one computation as input for another or the reusing of executables.

Referenced data has not be explicitly copied, but can be fetched from the original source if needed. The input and result references could be used to trace computations. It is also possible to find executions that use the same executable, but with different input data. Workflows are not explicitly specified, but a result of data reuse. Although, any existing workflow manager can be used to specify a worklfow on top of this transaction model. Making the approach technology agnostic.

3.1.4 Reproducible

Each execution request transaction can be re-executed if necessary. There is no difference between a first time execution and consecutive ones, except that the result is not included in a block. This property makes all computations inside a blockchain reproducible. The meta data, and the referenced data, encapsulated in transactions and verified in blocks are adequate. After the re-execution the results can be verified against results computed earlier and verifed by the network.

3.1.5 Reusable

Another scientific principle is the reusability of computed results, but also developed executables and existing input data. Reusability allows to build on top of new results and achievements. As described in section 3.1.3 all data can be reused by referencing it.

Chapter 4

System of Blockchains

In comparison to other blockchain projects and the reference blockchain Bitcoin, the here presented approach comes with build-in sidechain [Bac+14] support, but without the need of a main blockchain and without the need to handle assets exchanges. The main motivation behind the splitting of scientific, or general, computations into smaller units is the project structure of such computations. Results and other data are shared between few projects or the whole field, but most times are not reusable in other fields. With that in mind it makes sense to encapsulate a set of computations, with related workflows, in independent blockchains. To keep the specification future proof and to facility sharing of data between blockchains, by referencing cross blockchain transactions, the transactions come with build-in support for references in external blockchains.

Allowing references from one blockchain to another, as build-in mechanism, allows sharing of results, but also executables and input data, across projects. Additional it allows each project to maintain and release customized blockchains.

+ Advantages

Solves Scalability Problem Maintaining smaller blockchains solves the scalability problem during block generation. Consensus can be easier reached in these smaller overlay networks.

Independence By allowing each project and/or organization to maintain an own blockchain, there is no need to maintain a shared overlay network. Additional it allows private blockchains for specific scenarios, where the data can not be shared, e.g. because of privacy or regulatory issues.

Interoperability By design the blockchain structure and protocol supports cross-blockchain references, in order to not compromise the scientific principles of reproducible, verifiable, reusable and transparent computations.

Referencable Each entity inside the blockchain, e.g. input data, executables and results, are not only referable by other blockchains, but also from papers and other publications. The unique URI to the data can be used to refer to a data entry, but also to a complete workflow, empowering executable scientific computation as presented in [Str+11].

Smaller blockchains size Instead of downloading and managing a big global blockchain, with all transactions ever made, this approach allows to handle only the needed (own and referenced) blockchains. Old blockchains can be hence purged, without compromising the verifiable of new blockchains.

- Disadvantages

Blockchain management overhead The need to maintain multiple blockchains introduces, to some extend, an overhead. The fact that the blockchain management is build into a lower level protocol reduces this overhead to some extend, at least for application developers and researchers that are using the system.

4.1 Cross-Blockchain References

As mentioned in the previous chapters the idea behind this project is to make computations reproducible, verifiable, reusable and transparent. The encapsulation of computations in separated project or organization specific blockchains should be hidden inside the protocol and not be visible to the user. To maintain these properties it is necessary to have a build-in mechanism to share data references across blockchains. The same mechanism makes it possible to share results in scientific publications or any means appropriate, e.g. project website, posters, presentation slides.

4.1.1 Reference Specification

The following formal URI specification, in EBNF [Pat80], is used to reference fields inside transactions from the same or external blockchains. By using the *magicno*, the identifier of the blockchain, the transaction id and the referenced field, it is possible to uniquely reference each data entry.

⟨*URI*⟩ ::= ‘‘`blockchain:`’’ ⟨*magicNo*⟩ ⟨*trx*⟩

⟨*trx*⟩ ::= ‘‘`?trx=`’’ ⟨*trxid*⟩ [‘‘`#`’’ ⟨*field*⟩]

⟨*field*⟩ ::= ‘‘`inputDataRef`’’ | ‘‘`executableRef`’’ | ‘‘`resultId`’’ | ‘‘`resultDataRef`’’

It is also possible to only reference a single blockchain, e.g. for the initial sharing where the research partners can publish project related transactions, or a single transaction.

Examples with shortened magicno and trxid, normally a sha256 hash.

- Result reference:
  blockchain:01ba47...ca546b?trx=96dcec...9824cc#resultId
- Executable reference:
  blockchain:01ba47...ca546b?trx=96dcec...9824cc#executableRef
- Input reference:
  blockchain:01ba47...ca546b?trx=96dcec...9824cc#inputDataRef
- Specfic result reference:
  blockchain:01ba47...ca546b?trx=13a804...58c43a#resultDataRef

Cross-blockchain references inside transactions are automatically dereferenced by first checking if the blockchain is locally available, if not a DHT lookup is made. After fetching the blockchain or waiting until the referenced transaction is locally available, the field is extracted and the new reference is dereference the same way, or alternatively the data is fetched and used.

Not part of the reference implementation, but a feature needed for production use, is a user interface that allows to dereference links shared in publications and via other means. The steps are the same as described above, with the only difference that they are trigger manually by providing the link.

4.1.2 Hierarchical blockchain references

By introducing a generic transaction for transaction sharing across blockchains, it is possible to build hierarchical blockchains with a more abstract representation for higher blockchains in the hierarchy. In the scope of this domain specific blockchain the lowest blockchain keeps track of each execution with related flow. This representation is related to one specific project, e.g. a four year EU-funded research project. Main results

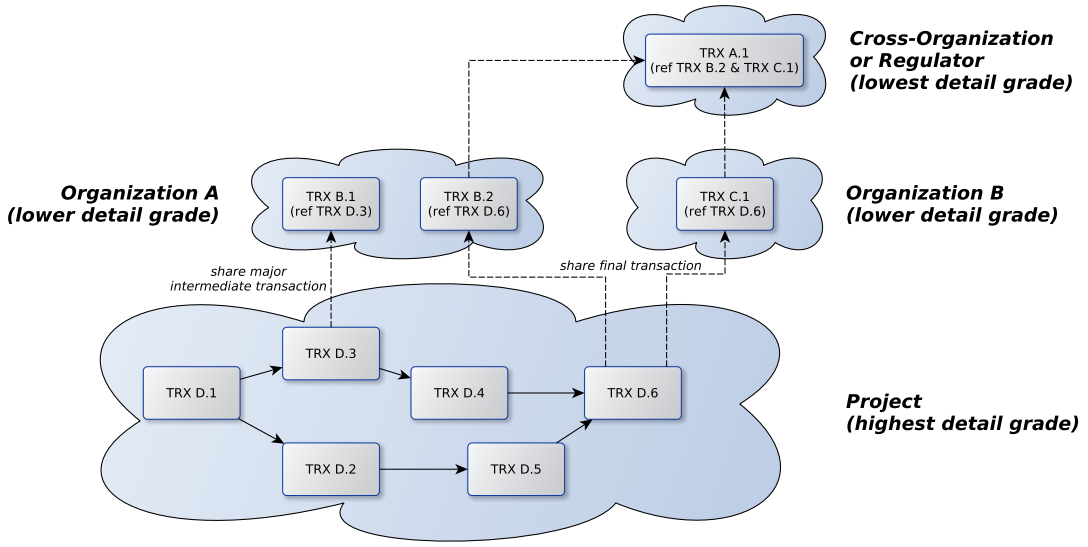


FIGURE 4.1: Example of hierarchical blockchains, with one shared project over two organizations.

are shared in a blockchain one level higher. This blockchain does not contain each intermediate result, but only relevant data for the organization or the broader field. Such a hierarchical structure makes it possible to have smaller blockchains, that keep track of relevant results and if necessary each result can be traced and verified by fetching blockchains one level deeper. An example of such a hierarchy is shown in Figure 4.1.

Additional to the technical advantage, to have smaller blockchains with only relevant data, this structure reflects also real world organizational structures, making the integration easier.

4.1.3 Accessing the overlay network of an unknown blockchain

Using independent, but linked, blockchains and the dynamic nature of the overlay networks that manage these blockchains, introduces the challenge of finding the right nodes. One solution to this problem is the encoding of entry point nodes inside the reference. From a theoretical point of view that would solve the problem, but makes the lookup unreliable and dependent on the specified entry point. What happens if this entry point, or even entry points, are not reachable? One solution, borrowed from file sharing technologies, is the use of a Distributed Hash Table, or DHT for short, to lookup nodes that control specific data, in this case blockchains. That solution not only scales well, but also adapts, in nearly realtime, to leaving and joining nodes and structure itself accordingly.

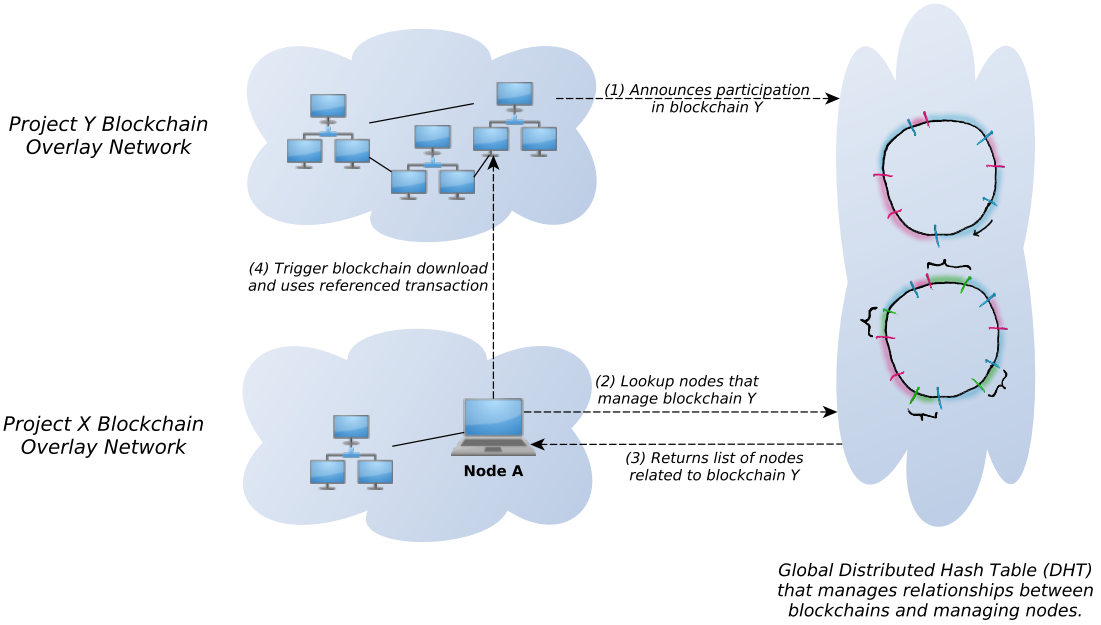


FIGURE 4.2: Cross blockchain lookup, if referenced blockchain not known

In Figure 4.2 a cross blockchain lookup is shown, on the example of two blockchains X and Y, from the perspective of *Node A*. In this example one node from blockchain Y announces participation to the global Distributed Hash Table. After that *Node A* receives an internal transaction in blockchain X that references a transaction in blockchain Y (not shown here). In order to be able to dereference the data, it has to fetch and verify the blockchain Y. By looking up managing node of blockchain Y in the DHT, *Node A* knows from where to fetch the blockchain.

4.2 Future Extensions

The here proposed hierarchical multi-blockchain approach can be extended in multiple ways. The inherent structure has some basic semantic meaning and can be used to build on top more sophisticated search mechanism. Without going into detail or proposing a first approach, it can be said that the higher levels expose a more abstract view, are hence faster searchable. Most times this view on the computations, that focuses only on the results is adequate enough. Although, if raw data or executables should be reused it is necessary to be capable to search over all computations ever made. This, much broader, search is more expensive and can be supported with technologies such a semantic triple stores [Roh+07], graph databases [Jia+07] or alternatives. A further simplification and optimization for search algorithms build on top is the introduction of

a computational blockckhain classification ontology, that classifies transactions during creation time.

The underlying trusted distributed data structure, that is shared via multiple parties, makes is possible to deploy multiple technologies and approaches on top, without breaking the underlying trusted core. Meaning that different parties can deploy different extensions, without breaking the interoperability.

The here proposed multi-blockchain approach, with related discovery mechanism and URI schema for web-compatible sharing capabilities, is only the minimal set to guarantee interoperability. It was developed with extensibility in mind, in order to increase the future proof nature, needed for such systems.

Chapter 5

Blockchain and Protocol Specification

5.1 Blockchain Specification

5.1.1 Transactions

One of the main concepts of a blockchain are transactions. This section defines the two transaction types, *ExecutionRequest* and *ExecutionInstance*. An *ExecutionRequest* is used to submit a new computation to the network. Additional to references to input data and executable, it also contains an identifier for the result. By introducing such a field, it is possible to reference the result, without waiting for the computation to finish. After executing a *ExecutionRequest* a new transaction of type *ExecutionInstance* is created. This transaction references the *ExecutionRequest* and includes also a reference to the result data. Only valid transactions, as computed by the majority of nodes, are confirmed by the network and added to a new block. A high-level view of both transactions and the relationships between them are visualized in Figure 5.1.

Fields only relevant for the serialization are not explicitly explained in this section, e.g. the type field or all length fields, but can be found in Appendix Table B.1 and B.2. These tables include a short description, the byte size of each field and all relevant information for an implementation.

Header Section

Both transaction types share the same header structure. The following list explains the most important fields and the functionality.

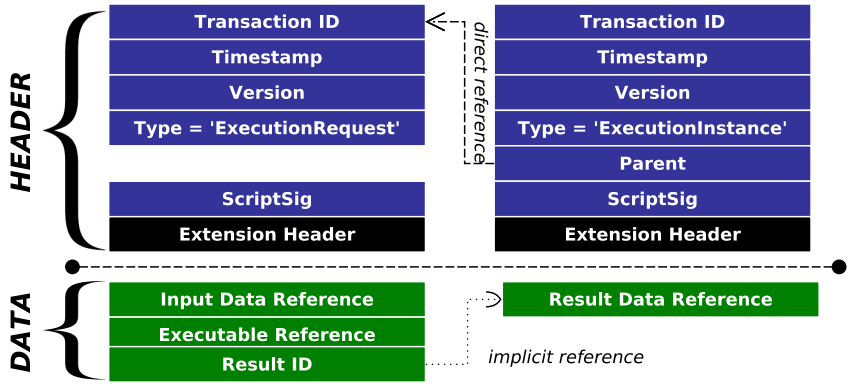


FIGURE 5.1: Visualization of the two, domain-specific, transaction types.

Transaction ID This field is used to uniquely identify the transaction and assures that it can not be modified. The ID is computed by hashing the hexadecimal representation of the rest of the transaction. As hashing algorithm a double SHA-256 is used.

Timestamp The time when the transaction was created.

Version The protocol version that was used to create the transaction. Used for stepwise roll-outs of new protocol versions, without breaking existing implementations.

Type Could be either *ExecutionRequest* or *ExecutionInstance*.

ScriptSig Extensible scripting field that is currently used for cryptographically signing the transactions. A signature has the following format "*$signature $publicKey* $OP\_CHECK\_SIGN\_BTC$" and uses the same format and algorithm as Bitcoin. The specified OP code constant allows the future switch to another algorithm. The signature links the transaction to an identity, that could be as simple as a cryptographic keypair or as complex as a strong identity, verified by an identity provider, and protected by smart contracts[1].

Data Section

Both transaction types have a different data section. An *ExecutionRequest* has the fields, *Input Data Reference*, *Executable Reference* and *Result ID*. In comparison an *ExecutionRequest* has only the field *Result Data Reference*.

All fields ending with the term *Reference* can have the following format:

[1]Smart contract backed strong identities are a pretty new concept and could be used complementary to this system, in order to increase trustworthiness in the system, by validating institutions and individuals involved in a project. The author of this paper works on such an identity layer.

Shared Data Storage Local reference to a remotely shared data storage, prefixed by file://$$SHARED_DATA_DIR$$/. Could be for example SSHFS, Amazon S3, SMB or any other remote mountable format.

Magnet Link URI scheme as used de facto standard in p2p filesharing, see http://magnet-uri.sourceforge.net/magnet-draft-overview.txt.

Internal Reference Cross-blockchain transaction reference as described in subsection 4.1.1

Other Future extensions are possible, as long as they are released and rolled out with a version update.

5.1.2 Block

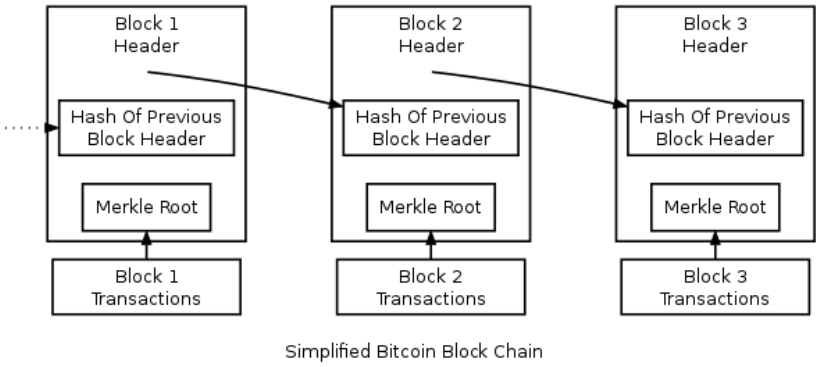


FIGURE 5.2: Simplified Blockchain structure and linking of blocks.
Source: http://bitcoin.org, Accessed: 25. November 2015

A *Block* is a collection of transactions with a unique order. Each block has exactly one predecessor and exactly one or none successor, dependent if it is the last generated block or not. Each block is cryptographically linked to all predecessor and a change in any block generated in the past, breaks all newer blocks (see Figure 5.2). These properties make older blocks, and hence the containing transactions, tamper-resistant. By including cryptographic proof of referenced files, hashing of file content, it is possible to detect changes in external referenced files. The resulting blockchain acts as proof of execution, but also protects results from changes. Additional it allows to trace back execution flows to the beginning and have a global overview about all computations ever performed. Per definition a transaction is verified if it is part of a valid block that is part of the blockchain, or formulated otherwise, if the majority agrees to include that block in the agreed blockchain.

5.1.2.1 Block Generation

The network has to agree on the next block, but also on a uniquely ordered transaction list. If the majority of the network accept the next block and uses it as reference for a successor block the included transactions are confirmed.

Purposely, this work does not limit the choice of a consensus algorithm. Dependent on the network size that could be something like a modified version of the Byzantine Generals Problem algorithm or alternatively the Proof of Work of Bitcoin, if computation power could be spared and nodes can be incentivized to participate. The first part is assumed to be false, the latter true.

The developed multi-blockchain approach allows to have smaller overlay networks that are responsible for the block generation. In combination with cross- blockchain references global interoperability is assured. Breaking the consensus problem down to a size that is managable by most well-tested algorithms, e.g. Paxos [Lam05] and similiar.

The proposed framework allows also to substitute the block generation aka. conensus algorithm rather easily, assuming that the majority of the blockchain nodes agree on the change.

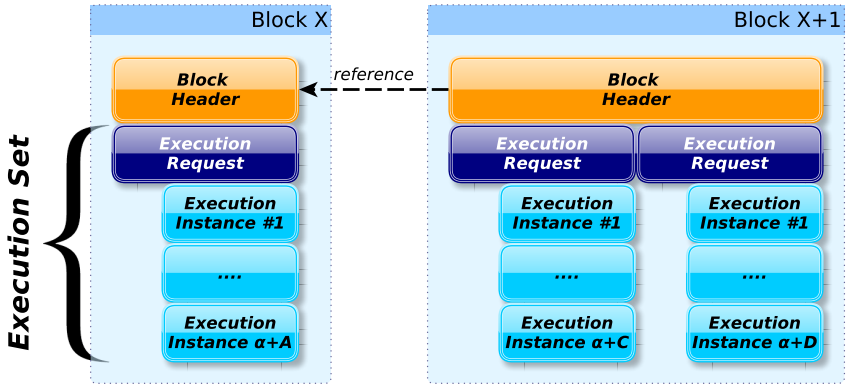


FIGURE 5.3: Transaction Sets inside a Block.

Important for the block generation, in the context of the scientific computation engine, is the fact that all *ExecutionInstances* have to be part of the same block as the related *ExecutionRequest*, see Figure 5.3. This allows the node that generates this block to verify if the majority of the nodes have computed the same result and only then add all transactions to the new block. If that is not the case, e.g. if every node has computed a different result, the computation transactions are rejected. Majority means at least $50\% + 1$.

When are ExecutionRequests executed and how does that relate to the block generation?

There are two potential ways when a node can execute a received *ExecutionRequest*, (a) when a new transaction is received, that is not yet part of a block or (b) when a new block is received, that includes one or more *ExecutionRequest* transactions. In order to have an additional check of the validity of an *ExecutionRequest* with related *ExecutionInstances*, computations are immediately executed when a new transaction is received. That allows to verify only valid *ExecutionRequest* with related *ExectionInstances* if the majority of nodes have published the same results, otherwise all related transactions are ignored by the network. A positive side-effect of this design choice is that one block contains a complete computation with request and results. That makes not only conceptional sense, but improves also performance. A node that receives a new block can be sure that all relevant information are part of this block, without waiting for potentially new results. If nevertheless a new computation has to be performed a new *ExecutionRequest* transaction can be created.

Formalization of the steps from a single node perspective:

1. The node receives a new *ExecutionRequest* transactions.
2. The node checks if it has related *ExecutionInstance* transactions.
   Through network delays and time that it takes to propagate a transaction it is possible that computational results are received before the actual request).
   (a) Number of known *ExecutionInstances* **is lower than** the threshold value α:
   Execute the *ExecutionRequest* and propagate the results in a new *ExecutionInstance* to all known nodes.
   (b) Number of known *ExecutionInstances* **is higher or equal than** the threshold value α:
   If this transaction set (request + related instances) is not yet part of a block generate a new block.
3. The node responds to each new block request of the same blockchain and that reference the previous block with the new block.

5.1.3 Scripting Language

The script field *ScriptSig* inside a transaction specifies how the transaction should be validated. The current version only accepts the following format, but is extendible for future more complex scenarios.

$signature $address **OP_CHECK_SIGN_BTC**

For compatibility and the possibility to re-use the intensive work done by the open source community the keypair uses the same cryptographic algorithm as Bitcoin does. It uses the Elliptic Curve Digital Signature Algorithm (ECDSA) [JMV01]), with the elliptic curve secp256k1, and the shortened version of the public key also known as *address*. The *$address* in this context looks the same as a Bitcoin address and is derived from the public key. The *$signature* is computed signature of the hexadecimal representation of the binary transaction (excluding the ScriptSig field), made with the related private key. The *OP_CHECK_SIGN_BTC* defines the action and the keypair format and is used to differ between current and future implementations. Such a structure makes it also possible to use different cryptographic algorithm at the same time and in the same blockchain.

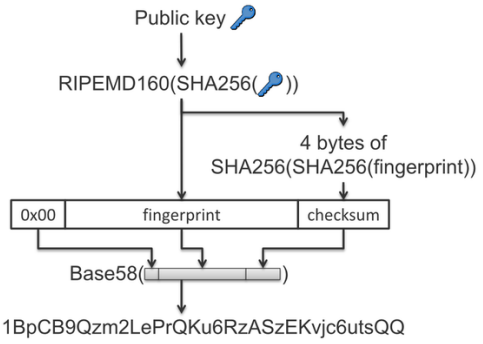


FIGURE 5.4: Shortened address derivation from public key.
Source: http://en.bitcoinwiki.org/Bitcoin_address, Accessed: 25. November 2015

The derivation of the address from the public key is shown in Figure 5.4. The shortened version is easier to share and manually to validate.

5.2 Protocol Specification

Message are exchanged via TCP in raw binary format. The internal representation of the message with related payload is transformed to a byte sequence in big endian notations. The following table specifies the format of a raw package send between nodes over the wire.

A node that received a package, as specified in Table 5.1 always performs the following generic steps:

1. De-serialize the raw binary package

	MagicNo	**Command**	**PayloadSize**	**Checksum**	**Payload**
Size (Byte)	32	1	4	4	*dynamic*
Description	Blockchain ID	Command flag (see below)	Size of payload in bytes	Payload checksum	Content to transmit
Creation	Double SHA-256 of UUID	Predefined commands	Size in bytes of the payload	First 4 Bytes of hash checksum of the payload content (hex representation of serialized version)	The payload that has to be transmitted. Differs based on command, e.g. transaction or block
	MagicNo	**Command**	**PayloadSize**	**Checksum**	**Payload**

Table 5.1: Specification of a serialized package.

2. Compute the checksum of the payload (*sha256(sha256(payload))*) and validate the first four bytes against the provided checksum field. This validation is needed to detect compromised payloads, e.g. because of transmission errors.

3. De-serialize the payload if necessary, e.g. to a Transaction or Block

4. Execute the command related logic

Command Name	**CMD Flag**	**Description**
VERSION	0x00	Initiates handshake and it is the first package send to a new node
VERSION_ACK	0x01	Response to the VERSION package and finalizing of the handshake
REQUEST_BLOCK	0x02	Periodical message to request a newer block
BLOCK	0x04	Response to REQUEST_BLOCK request
TRX	0x08	Propagation of a transaction
CONFIRM_TRX	0x10	Confirmation that transaction was received
GET_PEERS	0x20	Request all known nodes, related to a specific blockchain, from a node
PEERS	0x40	Response to the GET_PEERS request with a list of known nodes

Table 5.2: Specification of all package types available in the protocol.

5.2.0.1 Example: Serialized VERSION package

```
0000: 9595 c9df 9007 5148 eb06 8603 65df 3358  ..I_..QHk...e_3X
0010: 4b75 bff7 82a5 10c6 cd48 83a4 1983 3d50  Ku?w.%.FMH.$..=P
0020: 0000 0000 146b c383 fdff ffff ff00 0000  .....kC.}.......
0030: 0000 0000 0041 d553 2c1b 8000 00         .....AUS,....

Message Header:
  9595 c9df 9007 5148 eb06 8603 65df 3358
  4b75 bff7 82a5 10c6 cd48 83a4 1983 3d50 - Blockchain MagicNo

  00                                      - VERSION command flag 0x00
  0000 0014                               - Payload size; In this case 20 Bytes.
  6bc3 83fd                               - 4 Byte Checksum of Payload

Version Payload:
  ffff ffff                               - Protocol version of the node
  0000 0000 0000 0000                     - Supported services
  41d5 532d 9640 0000                     - Unix timestamp. In this case:
                                            Fri May 08 2015 15:12:57 GMT+0200 (CEST)
```

5.2.0.2 Example: Serialized VERSION_ACK package

VERSION_ACK has the same structure as the VERSION message, except the change of the command flag from *0x00* to *0x01*.

5.2.1 Inter Node Communication

The following section describes the interaction of a node with the network. The first time a node joins the network it bootstraps itself by connecting to a specific bootstrap server or any other node in the network. The bootstrap server is not needed and introduces a single point of failure. Nevertheless it is used until a bigger network with stable nodes is available.

Definition of 'Boostrap Server' A boostrap server is a node that does not host any blockchain, nor any business logic, it only provides an entry point to the Distributed Hash Table (DHT), containing references from magicNo (blockchain IDs) to node IPs that manage this blockchain.

lookup(*$magicNo*) Accepts lookup calls with the magicNo as input parameter and returns a list of all nodes. Each entry reflects an enpoints in the form of IP:Port where the blockchain handler that manages that specific blockchain runs.

announce(*$magicNo*, *$IP*, *$Port*) Accepts announce messages from nodes. An announce message means that the specified node runs a blockchain handler on the specified port and IP, that manages the blockchain with the specified magicNo.

Definition of 'Node' A participant in an overlay network that manages at least one blockchain and actively participates, by verifying and forwarding transactions and querying for new blocks.

Defintion of 'Peer' A remote node that has an active connection with the local node. More specifically the handshake was successfully completed and if the Peer is online it will be queried for new blocks and new transactions are send to it.

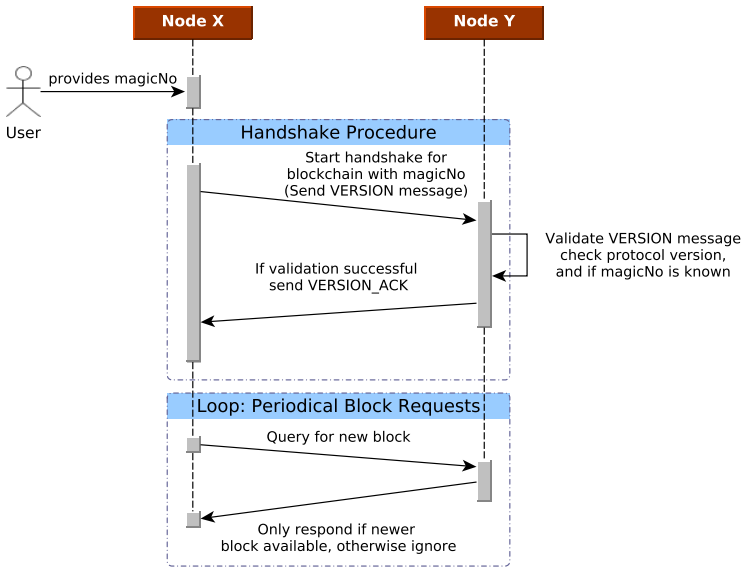


FIGURE 5.5: First time handshake to remote node and periodical block requests.

The following steps describe the actions that have to be performed by a single node, in order to be part of the network. These steps are executed for all known blockchains independently. A node can only participate in the blockchain ecosystem if at least one blockchain magicNo is known. As described in the previoius chapters, the blockchain magicNo could be retrieved from a scientific paper, where it was shared as part of a computation or the research project. It could be also shared in private, e.g. by e-mail or instant messaging.

1. Check how many active connections to nodes with the same blockchain are open (foundNodes).
   (a) If foundNodes >= MAX_PEER_CONNECTIONS
   Nothing to do.
   (b) If foundNodes < MAX_PEER_CONNECTIONS
   Continue with step 2
2. Find inactive peers in the local database and try to connect to them. If none or not enough peers are found a DHT lookup with the current magicNo is triggered, as shown in Figure 4.2.
3. Connect to the newly found or previously known and now online peers and start periodical block requests. Figure 5.5 shows the first time handshake and the periodical block request calls.

Additional to executing these steps for each blockchain, they are also executed periodically in a random time interval. The default of this interval is between 500 milliseconds and 4 seconds. Randomness in the interval distributes the load if the node manages multipe blockchains. Dependent on the number of blockchains (more blockchains means a longer time interval) and the available hardware (how many requests can be executed) these values can be fine tuned, making the overlay networks more dynamic and tailored to the needs of specific nodes.

Chapter 6

Implementation

6.1 Software Environment

In order to guarantee optimal performance and a simple code base each part of the software stack was chosen with the task at hand in mind. This section justifies each major decision.

Before going into details the overall context has to be specified. Most importantly the fact that the software should run on off-the-shelf computers, e.g. laptops and desktops. That limits the choices by eliminating heavy weight server software, in favour of of light weight ones. Nevertheless the implementation needs to be capable to handle multiple connections in parallel, for exchange of blocks and transactions, with potentially multiple blockchains.

All here mentioned components are well tested and popular in industry, making the design suitable for a production ready system.

NodeJS (https://nodejs.org/) as event-driven, non-blocking I/O programming language is best suitable to handle the communication between nodes, but also internally between components. The whole implementation is event driven and actions are triggered by the users or by an incoming package from the network.

Redis (http://redis.io/) is used for inter component communication. Components are loosely coupled via publish/subscribe channels. This allows substitution and extension of the software during runtime. Additional it allows to have multiple components that handle the same event, but in a different way, e.g. a new transaction triggers a computation, but also notifies the user.

Docker (https://www.docker.com/) was chosen as virtualization tool for the execution environment, but also for during development to simulate an overlay network. The compromise between lightweight, speed and portability makes it the perfect choice to encapsulate execution and run multiple nodes on the same machine. By deploying the containers executions are run on each node inside the same environment, avoiding dependency and version problems.

MongoDB (https://www.mongodb.org/) Not as light weight as the other components, but suitable for the storage of transactions and blocks in BSON representation. MongoDB is a document based NoSQL database, that stores files in a JSON similar format, that supports also binary data. The powerful querying capabilities allow the extension of a future single node search engine.

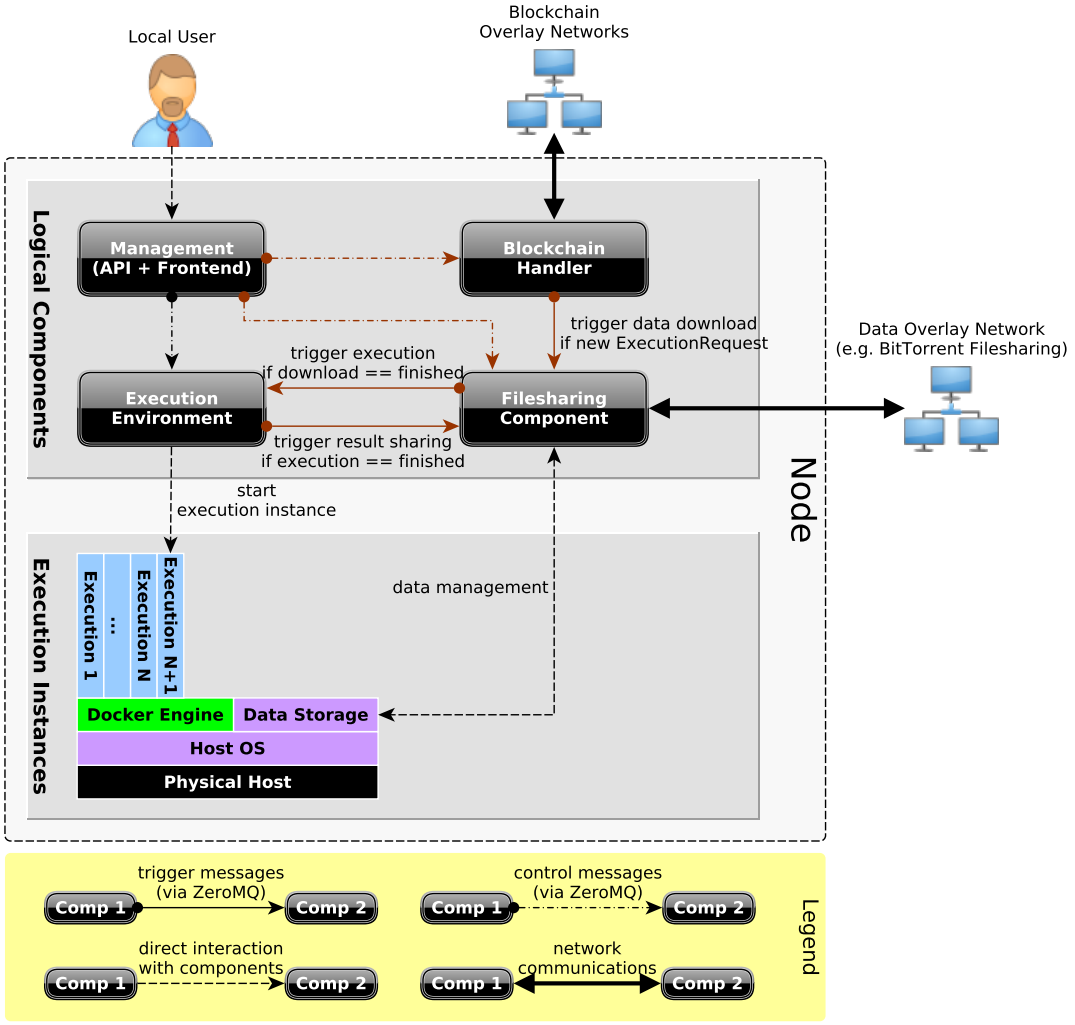


FIGURE 6.1: High Level Architecture with inter component communication and execution environment design. ZeroMQ parts were substituted at a later stage with Redis publish/subscribe channels, for extensibility. Conceptional the flow is similar to the one shown here.

Figure 6.1 depicts a first architecture design. It defines needed components and their interaction with the two conceptional overlay networks, one for blockchains and one for raw data. The initial design of the internal communication was done via ZeroMQ sockets (`http://zeromq.org`), but later one change to Redis publish/subscribe channels. The change was made in favour of a simpler structure and easier extensibility, e.g. extension with new modules, without changing the underlying communication flow. In Figure 6.2 these channels and the resulting flow is shown. The extracted framework, that is reusable for other blockchain architectures for other domains, can be seen in Figure A.1.

6.2 Event-Driven Architecture

The complete architecture is event-driven and reacts on events trigger by a local user, e.g. creating and propagating a new execution request, or by remote nodes, e.g. incoming blocks and transactions.

Internally events are communicated via Redis publish/subscribe channels. In Figure 6.2 the internal communication between component is shown on the example of an incoming *ExecutionRequest* transaction. Not shown here, but possible is the listening of multiple components to the same channel. For example a block generation component will be listenting to the same *transactions* channel and decide if a new block should be generated or not. Another example is the subscription of a notification component to the *executions* channel. If an execution fails the user is notified via e-mail or any other means. This type of architecture plugs well into the blockchain protocol and makes it also easy extensible.

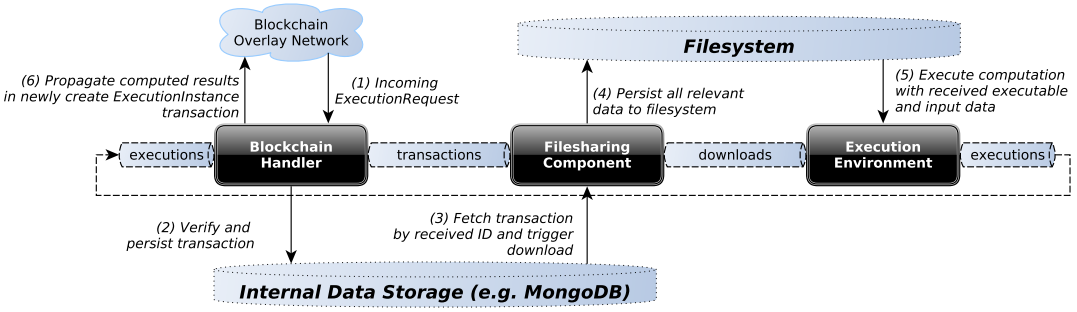


FIGURE 6.2: Incoming ExecutionRequest Scenario - Node receives a new transaction, triggers computation and propagates the result.

6.2.1 Publish/Subscribe Channels with Message Format

Table 6.1 describes the most important and current available inter-component publish/subscribe channels in the reference implementation.

6.3 Development Environment

Distributed systems are notorious difficult to develop and to debug, that is one of the reasons why docker containers with a shared database and redis server was used. Each node has an unique identifier, encapsulating the database and publish/subscribe channel names (attached to the end of the name, e.g. transactions_*$NODEID*). This specific setup can be run completely locally by creating a virtual network.

The following steps explain the setup and start of the development environment. All steps are executed via *./build_virtual_network.sh X*, where X is the number of nodes that should be started.

1. Step: Build a generic docker image that includes all components and exposes the default port 6431 to the host.
2. Step: Build and start a generic MongodDB and Redis container for each group of 5 nodes.
3. Step: Start the number of nodes as specified by as input parameter. Each node performs a setup step during first time start. During this setup phase a new node ID and a keypair is generated.

The node ID is only relevant for the development environment and is not necessary for a production environment where only one node per machine is started or where MongoDB and Redis are not shared.

Channel	***transactions***
Description	All incoming and new transactions are published to this channel.
Producer	Blockchain Handler
Consumer(s)	Filesharing Component, ...
Message	{ magicNo: $magicNo, id: $id, type: $trxType }

Channel	***blocks***
Description	All incoming and new blocks are published to this channel.
Producer	Blockchain Handler
Consumer(s)	...
Message	{ magicNo: $magicNo, id: $id }

Channel	***downloads***
Description	Status of data download, most importantly when ready to use.
Producer	Filesharing Component
Consumer(s)	Execution Environment, ...
Message	{ status: $status, magicNo: $magicNo, id: $id, inputData: $inputData, executable: $executable, resultPath: $resultPath }

Channel	***executions***
Description	Status of executions, most importantly when execution is finished and results are ready to be published.
Producer	Execution Environment
Consumer(s)	Blockchain Handler, ...
Message	{ magicNo: $magicNo, id: $id, returnCode: $returnCode, resultPath: $resultPath, resultFile: $resultFile, errorFile: $errorFile, resultDataRef: $resultDataRef }

TABLE 6.1: Publish/subscribe channels as used in the reference implementation.

Additional to the node ID a keypair is generated. For compatibility and convenience (short form) this keypair has the same format as bitcoin addresses. This keypair is used to sign all created transactions and blocks.

A positive side-effect of the use of docker containers is the easy deployment of containers to a production system. Participating in the scientific computation engine is then only a matter of deploying a docker container.

Chapter 7

Results and Discussion

The here presented approach proposes a new solution to solve the reproducibility, verifiability, reusability and transparency of computations. Instead of building another framework for computations, the meta data that describe an execution is de-capsulated from the real computation. All needed data, like executables, input and output data, are referenced and made tamper resistant by introducing cryptographic mechanism, like transaction signing with asymmetric keys and hashing of referenced files. The non-obtrusive nature of the resulting solution, combined with secured meta data and the introduction of an URI scheme for easy sharing, satisfies all of the above mentioned properties. Additionally, the integration in existing computation frameworks is made as easy as possible.

Lets analyse the mentioned properties and how the proposed solution empowers them. More details were given in the previous chapters.

Reproducibility and Traceability Per definition a blockchain is a distributed ledger where each node stores all transactions ever made. By having access to the complete meta data of all computations, it is possible to execute a past computation by using exactly the same transaction that was originally used. Furthermore, it is possible to run multiple executions by following the references between the transactions.

Verifiability Each transaction is verifiable on its own and in the scope of the whole blockchain. By using multiple layers of cryptography single executions, but also computations consisting of multiple steps, can be verified by any involved party.

Reusability A well defined cross-blockchain URI schema allows the referencing of transaction and moreover fields inside a transaction. That could be input data, executables or results. The steps needed for executing a past computation are exactly the same as the first execution. Proposed extensions to this approach, like a scientific marketplace, increases reusability by increasing usability.

7.1 Fine-tuning for different use cases

In order to be able to compare use cases and how different changes effect performance and quality metrics, two base cases are given. The first scenario is tuned for performance and does not at more

value to current computation frameworks. In comparison the second one is the other extreme, has lower performance, but is more resilliant against malicious nodes and error.

7.1.1 [Base Case #1] Tuned for Performance

The base case that is used as reference for all other use cases. It establishes a baseline by adding the distributed ledger as parallel data structure that keeps track of executed computations. This base case does not satisfy the properties of reproducible, verifiable, or only in a limited way.

All transparency values are at a scale from 0 (worst) to 1 (best) and chosen from an outsider point of view, a party that has not participate in the original computations. For the performance zero means worst performance and one highest.

Bootstrapping One central DHT bootstrap server that is used by every node that wants to join the overlay network.

Overlay Network Private overlay network, only accessible by participating parties. All nodes are trusted and verified.

Performance High performance, without intercepting or slowing down the original computations. (Value: 1)

Transparency All values are 0.1, and not 0, because they are satisfied for the participating parties, but not for the overall scientific community.

Reproducible Value: *0.1*

Verifiable Value: *0.1*

Reusable Value: *0.1*

7.1.2 [Base Case #2] Tuned for Transparency

This use case is the other extreme. Nodes are not trusted, because everybody can theoretically participate in the network. Similar to the open structure of Bitcoin, it introduces bad actors, but also a more transparent and self-regulating environment.

Bootstrapping If at least one active node in the network is known it can be used for bootstrapping. If no node is known a fallback bootstrapping server exists.

Overlay Network Public overlay network, everybody can participate. Nodes are not trusted and decisions are made via distributed consensus algorithm.

Performance Lower performance as legacy system, because computations have to be performed multiple time, in order to guarantee there correctness and prevent bad actors to inject wrong results. The consensus algorithm adds another performance slowdown to the mix. (Value: 0.1)

Transparancy All results, with all intermediate steps, are accessible by everybody and can be verified, reproduced and reused without limitations.

Reproducible Value: *1*

Verifiable Value: *1*

Reusable Value: *1*

7.1.3 Discussion: How to decide for what to optimize?

The two base cases point out the two extreme ends. The first one (7.1.1) resembles mainly the current status quo. The proposed solution does only support the involved parties and does not add much more value. The same functionality can be achieved with current workflow management framework. In comparison the second use case (7.1.2) makes scientific computations accessible for the general public and other researchers. By building in mechanism to weaken bad actors, e.g. redundant computations and majority votes, trust in the overall system and the confirmed results could be increased. Even more the confirmation of computations can be seen as peer-review process. By crowd-sourcing the confirmation, experts can double check the computations and increase the overall quality even further. The downside of this approach is the decreased performance. Even if the computations are confirmed automatically, they have to be executed multiple times and the network has to agree on the order of confirmation, to have a global consistent state. The latter is not as big of a problem, because it can be performed in an asynchronous way, without intercepting with on-going or future computations. Although the former may be a limiting factor, especially when computations are intensive and can not be easily reproduced or split up into smaller junks.

The solution to this optimization problem is a hybrid approach. As outlined in this work, the proposed system is capable to handle multiple blockchains at the same time. That means a closed, but trusted, blockchain could be used for computations that have not to be confirmed or can not be executed multiple times. At the same time an open blockchain could be used for smaller scientific computations. A node can access and verify both blockchains, it can also reuse and reproduce results, but only actively participate in the latter. Dependent on the use, it is also possible that multiple blockchains build a hierarchy, as shown in Figure 4.1. This hierarchy allows to share only parts of the results in higher up blockchains, increasing performance by a smaller and easier to maintain structure, but without sacrificing traceability and verifiability. If necessary a node can fetch all blockchains of the hierarchy and trace the full computation, but it can also only use the higher level results.

7.2 Verifiable and tamper-resistance

Transactions, and hence computations, are verifiable and tamper-resistance per design. The distribute data structure allows each node to verify transactions. The following points explain the design decisions made and how they support the verifiability and tamper-resistance.

1. Point: The distributed data structure makes it harder to tamper with it, because every participating party has an identical copy.
2. Point: ScriptSig with signature and public key is a cryptographically strong means to protect transactions and proof that the content was (a) not changed by anybody and (b) was really created by the party that claims to have created it.
3. Point: Weaker protection against transfer errors and alike, is the hashing of the transaction and using the resulting hash as ID.
4. Point: The transaction ID is used for inter-transaction references, making it even harder to change transactions. If a reference transaction is changed the reference to it automatically breaks (ID changes).

5. Point: Combination of transactions inside a block. Blocks build a chain with references to previous and next blocks. References are created based on hashed content. If the content changes, the reference breaks.

7.3 Sharing of scientific computations

The development of a distributed and tamper resistant data structure is not enough to satisfy the use case of scientific computations. Another important property is the availability of a mechanism to share computed results. By introducing a URI schema and by having a global Distributed Hash Table (DHT), it is possible to look up and fetch any public available blockchain. The URI schema is similar to HTTP and the global DHT keeps in real time track who is currently online and manages a specific blockchain. Referencing one specific computation is only a matter of combining the blockchain ID (magicNo) with an ExecutionRequest transaction ID and sharing that link. The support for data sharing is part of the original design.

7.4 Performance and Reliability

The prototype implementation allows to fine tune different values. Dependent on the use case and the node context, the right parameters can increase performance for the single node, but also for the overall network. The following section describes each parameter and how it affects the behaviour.

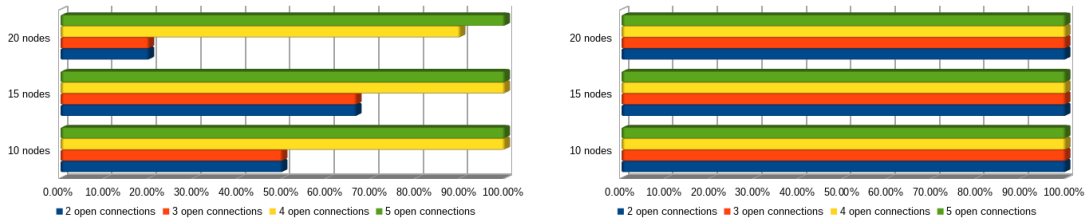
Packages, consisting of a fixed size header and the payload, are exchange via TCP in binary representation. The following values allow to fine tune connection parameters to optimize for the current environment. One node can manage multiple smaller blockchains or fewer, but bigger ones, or anything in between.

TIMEOUT *Default: 3 seconds* - Synchronous connections, like the handshake procedure or the propagation of transactions, accept a response from the other node. If no response is received the connection timeouts in the amount specified in the TIMEOUT variable. Faster networks should have a lower value, in order to be able to detect broken connections earlier and close them as soon as possible. Resulting in the possibility to open faster new connections.

TICK_LOWER and TICK_UPPER *Default: 9 and 10 seconds* A time interval from which a random value is chosen. This time interval marks the time between connection checks. A randomly selected time interval assures that, also if all nodes are started at the exact time, they do not perform at the same time the handshake procedure. Randomness spreads the global network load and minimizes peak times, e.g. during boot times or if multiple nodes join or leave at the same time.

BLOCK_REQUEST_LOWER and BLOCK_REQUEST_UPPER *Default: 3 to 4 seconds* Time interval for random value that indicates the waiting time between block requests. The same way TICK_LOWER and TICK_UPPER is used, also these values are chosen randomly for each iteration. A different random waiting time for each node minimizes peak requests, not only for the single node, but if each node behaves the same way also for the whole network.

BLOCK_REQUEST_TIMES After this amount of requests, block requests are stopped and during the next connection checking cycle (see “TICK_LOWER and TICK_UPPER”) randomly a new



(A) Static Overlay Network (B) Dynamic Overlay Network

FIGURE 7.1: Total number of nodes that received the propgated transaction.

node is selected. Additional to distributing load between nodes and switching them periodically, it also prevents nodes to be stuck with malicious nodes.

MAX_PEER_CONNECTIONS Maximum open connections at the same time per blockchain. If a node manages multiple blockchains at the same time this value should be lower. A higher value improves network propagation performance by receiving transactions and blocks faster (less hops between nodes if the connection degree is higher).

7.4.1 Transaction propagation

Transaction propagation is import for two reasons. First it assures that enough nodes receive an *ExecutionRequest* transaction and the minimum amount of executions can be reached. Second the node, or nodes, responsible for block generation have to receive the transactions too. This point could be solved for smaller, permissioned blockchains by actively propagating transactions to a selected subset of nodes responsible for this task.

	Maximum Open Connections			
	2	**3**	**4**	**5**
10 nodes	5 (50%)	5 (50%)	10 (100%)	10 (100%)
15 nodes	10 (67%)	10 (67%)	15 (100%)	15 (100%)
20 nodes	4 (20%)	4 (20%)	18 (90%)	20 (100%)

TABLE 7.1: Total number of nodes that receive the propagated transaction.

One requirement of the overlay network of a blockchain is that the graph depicting the network is connected. That means no node is unreachable. During the evaluation phase with a first implementation this property was not satisfied under all circumstances. In Table 7.1 and Figure 7.1a these results are shown. Most notable is that if the maximum open connections value is lowered and the total number of nodes is increased the number of nodes that receive a propagated transaction decreases. In Figure 7.1b it is shown that 100% propagation could be reached if nodes are selected randomly afer some time and transactions are propgated to all, and not only the current open, nodes.

The total number of nodes and maximum open connections per node reflect a smaller blockchain, as could be expected for a research project executed by a consortium.

7.4.2 Choosing right consensus algorithm

As described in chapter 2 the biggest challenge is the choice of right consensus algorithm. The first blockchain, Bitcoin, uses expensive computations to proof that work was done, to generate new assets

(Bitcoin) and encapsulate transactions inside blocks. This approach is not suitable, because it wastes too much computation power for block generations, that can be used instead for scientific computations. Additionally, it assumes that the network is untrusted, that is not true for research done by a trusted consortium. Another point against Proof of Work is the fact that the here presented blockchains are only a distributed ledger of transactions, without non-copyable assets that have to be protected. Alternative blockchain consensus algorithm are at this stage not mature enough to be considered. More research and deployment in a real world context is needed to finally decided if they are a better fit or not.

Consensus algorithms presented here as pre-blockchain approaches (see subsection 2.2.1) can be used for blockchains that agree on a trusted subset of nodes and are responsible for the block generation. Instead of assuming that every node can theoretically create new blocks and that the block generation has to scale at internet scale. It is assumed that research projects with a well-defined and trusted consortium can agree on a subset of nodes for this task. Nevertheless it allows external parties to access the computed results and reused them in the same, or any other, blockchain.

The here presented multi-blockchain approach, together with a sophisticated discovery mechanism, solves the consensus problem by breaking it down into smaller overlay networks. Allowing nodes to decided what blockchain overlay networks could be trusted and referencing transactions and fields across them, guaranteeing interoperability.

Chapter 8

Conclusion

In the scope of this work a solution to the reproducible computation problem was proposed. By using state-of-the-art (e.g. cryptography) and emerging (e.g. blockchain) technologies an alternative, non intrusive, but more trustworthy solution was designed. Furthermore, this work explains the first time how to develop a new domain specific blockchain in a systematic way and how it can be plugged into a hierarchical blockchain ecosystem.

The overhead of redundant computations, especially for big data computations, is for most industrial use case too expensive and inefficient, but for scientific computations it adds essential more value. The fact that no single entity has to be trusted and more importantly that scientific computations can be verified and checked independently. Additionally, the here proposed architecture results in a computation marketplace, where each computation can be traced back and easily shared via unique URI, the same way as we share today websites, creating a more open environment that empowers faster progress and innovations.

Future extensions and work include the integration of the here proposed solution into existing computation and data storage frameworks, but also into higher level workflow management solutions. An interesting future extension is the creation of a scientific computation marketplace on top of the available system. The hierarchical structure gives a first clustering of computations by projects, organizations, fields, etc. A next step can be the classification of single transactions by well-defined tags. By building a search engine and user interface on top of it, with an additional integration with a workflow manager, scientific computations can become more efficient by reusing off-the- shelf data and executables for computations. Everybody could then theoretically access the marketplace and create new computations by recombining them, e.g input data from a specific project combined with an executable from a completely different one, even from another discipline.

Figure 8.1 shows where such a scientific marketplace could fit into the current architecture. The Blockchain Stack and the underlying distributed ledger structure was developed in the scope of this work. The scientific marketplace can be build on top of it, as extension.

A more accessible scientific computation environment, with off-the-shelf data and executables, makes computations not only reproducible, verifiable, reusable and transparent, as the original goal was, but has also the potential to accelerate progress and innovation. The trend to involve the general public in

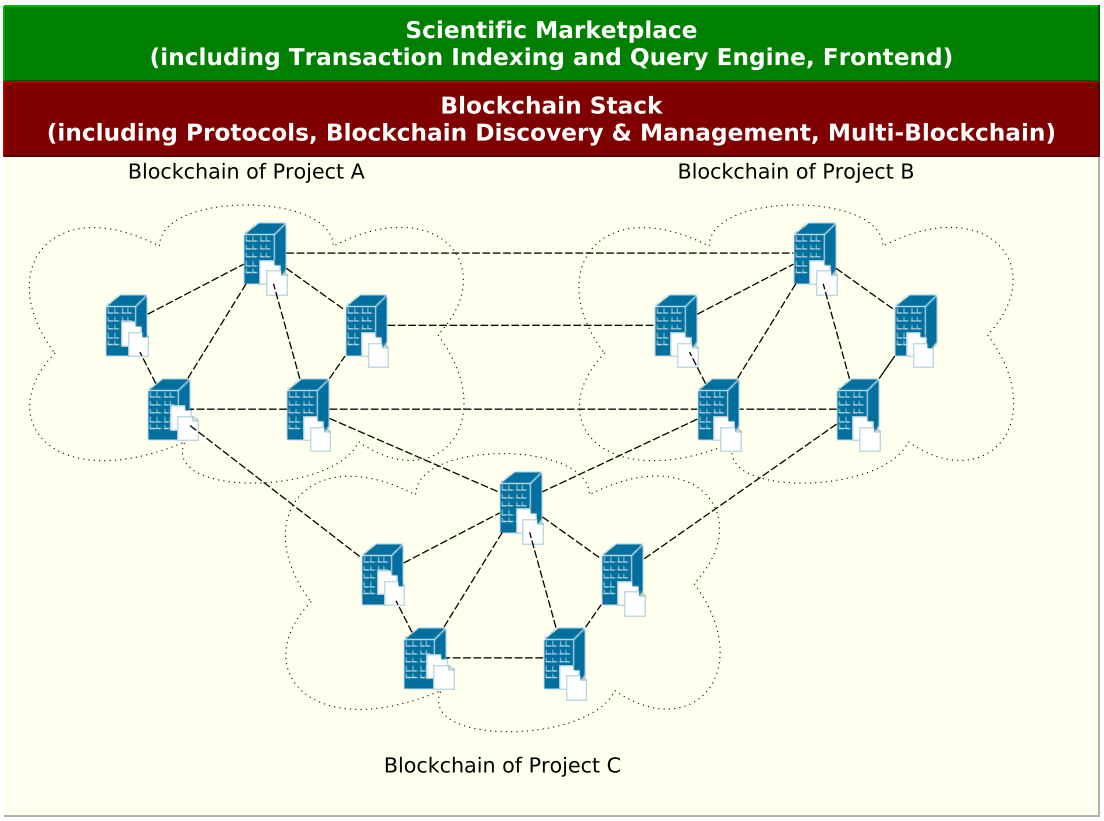


FIGURE 8.1: A Scientific Computation Marketplace as extension to the here proposed approach, for increased usability.

science can be seen in projects like SETI@home, ATLAS@home, CureCoin[1] and frameworks like BOINC [And04]. For now these efforts are mainly focused on using the raw computation power or crowd-sourcing a well-defined task, making it more accessible for non-experts. In comparison the resulting marketplace makes the underlying data and computation steps accessible and is more powerful, but assumes also some degree of expertise or willingness to experiment and explore. A positive side-effect could be the spark of interest in science and the increase in science from home.

The here proposed solution does not claim to be more efficient in perspective to performance, but it brings scientific principles back to digital computations and makes in the process science more accessible and more flexible. The scientific community has to ask itself if these principles are worth the decrease of overall performance.

The here designed framework, most notable the cross-blockchain references and the related blockchain discovery mechanism, are reusable for a multitude of use cases. As continuation of the started work in this paper, the author works on an identity layer based on the here designed and evaluated principles.

[1] CureCoin is a digital currency that combines Proof of Work with Folding@Home, see https://www.curecoin.net/, accessed: 5. July 2015

Appendix A

A Blueprint to design and develop new blockchains

This chapter introduces a blueprint how new blockchains could be developed. After discussions with experts in the field, researching public available blockchain projects and developing two internal blockchains (the here presented one and a Distributed Hosting Engine[1]), the here presented blueprint is a generic approach that can be reused to develop domain specific blockchains. By finding key components that every blockchain shares and by developing a modular framework, it is possible to use off-the-shelf components and only customizing domain specific parts. At the time of the writing there is no public known framework that allows to create new blockchain in a fast and painless way.

The here presented framework for blockchain applications must satisfy at least the following properties:

Modular By splitting the application into well defined components with a clear interface, it is possible to change parts, without touching the rest.

Loosely Coupled The rather complex nature of blockchain applications and the fast pace that new algorithm and approaches are introduced, makes it necessary to encapsulate components in such a way that they can be changed without touching any other part.

Well Documented In order to customize the here presented framework and to build on top of it, it is essential that all generic parts are well documented and a stepwise guide is given how to customize them.

Different software components: Blockchain Handler (key component) Filesharing Component (optional for all blockchains that reference files) Domain specific logic (one or more backend components that take care of domains specific parts).

[1] Distributed Hosting Engine http://blackgate.networld.to

A.1 Steps to develop a new blockchain

The following steps below show the non-trivial nature of developing a domain specific blockchain. It is a collection of useful questions to support the decision process. It should be seen as overall guideline. Some of the steps mentioned below can be executed in parallel.

1. Collect domain specific knowledge and investigate if a blockchain is the right technology. Answer for this the following questions.
   - For what purpose do you need a proof mechanism?
   - Why should that data be tamper resistant? Why should nobody be able to change historical information?
   - How does a transaction/block look like? What do you want to model?
   - What type of blockchain do you need? Is it permissioned/permission-less, public/private, ...?
   - What type of block generation aka. mining do you need? Is the deciding network small enough to use a classical consensus algorithm like [C+99] or do you need proof of work, proof of stake or any other approach?
   - Do you have one global blockchain or multiple smaller ones? If so, how are they linked? (Can be reused from this work)
2. Design a block and transaction specification based on the gathered domain knowledge.
3. Design a protocol specification. (Can be reused from this work!)

The inter-blockchain transaction referencing and the multi-layer protocol, developed in the scope of this paper, can be reused for the new blockchains. Making it possible to plug-in new blockchain, but only focusing on the domain- specific essentials, like how does a transaction look like and for what properties has the blockchain to be fine-tuned.

A.2 Framework Architecture

The here presented reference architecture can be used as starting point for blockchain applications. It introduces a highly modular and loosely coupled message oriented architecture. This type of architecture is useful for future modifications, without touching the existing components. Additional to be able to remove existing components, it is also possible to add new ones. The modularity allows to reuse existing off-the-shelf components and only build or modify domain specific ones.

A.2.1 Basic Component List

The following list outlines basic components, as identified during the work on different blockchain projects. Generic components can be reused and/or implemented in different ways, e.g. by switching the underlying algorithm.

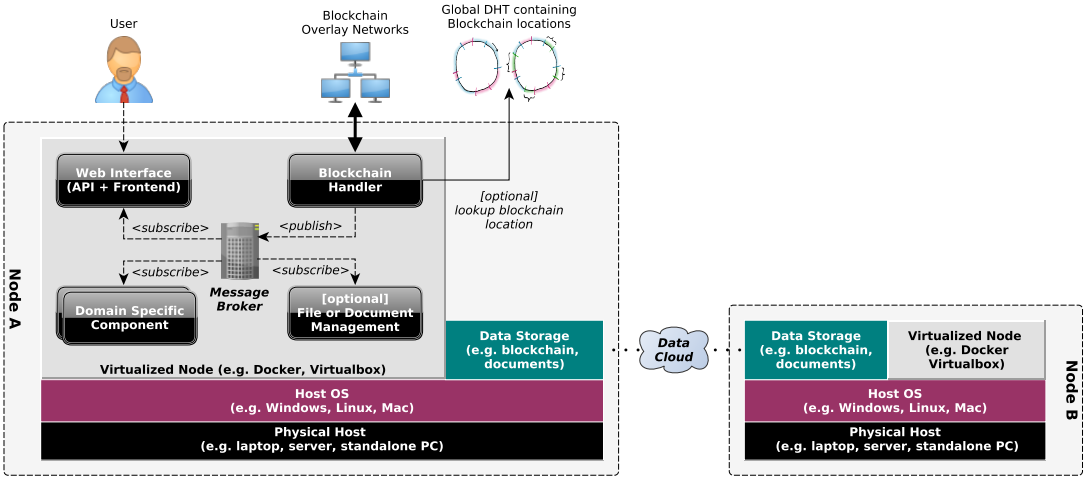


FIGURE A.1: Reference Architecture of the Framework

Blockchain Handler Each blockchain includes at least a component that implements the protocol and manages the blockchain. A basic protocol could be shared across different domain-specific blockchains, e.g. the transaction propagation and block querying, different block generation aka. consensus algorithm implementations and basic validation logic.

Web Interface (+ API) A generic blockchain viewer with basic functionality and API for third party integration could be reused. Functionality could include transaction submission, network statistics, ...

File or Document Management An optional file or document component could combine external data management with proof and tamper-resistance of blockchains. Such a component could be reused, not matter of the document or file semantics.

"Message Broker" Middleware that takes care of message passing in a publish/subscribe pattern. Could be Redis, RabbitMQ or similar.

Domain Specific Components These components are developed for a specific use case and could not be reused in an easy way. They are plugged into the framework by subscribing to one of the existing channels and optional creating a new channel for other components.

Appendix B

Binary Transaction Specification

Size	Description	Data Type	Comments
HEADER			
32 Byte	**TransactionID**	char[32]	Hexadecimal representation of double SHA-256 hash of the transaction (excluding Transaction ID field).
8 Byte	**Timestamp**	uint64	Unix timestamp: Seconds since 1. January 1970 (UTC) Note: Use 64 Bit to allow dates after 19. January 2038, see http://www.unixtimestamp.com/
4 Byte	**Version**	uint32	The specification version of this transaction. This field could be used to gradually upgrade the network to a new version.
1 Byte	**Type**	uint8	For *ExecutionRequest* 0x00 and for *ExeuctionInstance* 0x01. Extensible for future use.
4 Byte	**ScriptSigLen**	char	The length of the following ScriptSig field in Bytes.
? Byte	**ScriptSig**	char[?]	**TODO:** Figure out suitable scripting language.
4 Byte	**ExtHeaderLen**	uint64	The length of the following extension header. Default: 0. Used for future extensions.
? Byte	**ExtHeader**	char[?]	Future transaction header extensions, currently not used.
DATA SECTION			
4 Byte	**ResultDataRefLen**	char	The length of the following ResultDataRef field in Bytes.
? Byte	**ResultDataRef**	char[?]	Reference to the result data.

Table B.1: ExecutionInstance specification with field size and data type.

Field Size	Description	Data Type	Comments
HEADER			
32 Byte	**TransactionID**	char[32]	Hexadecimal representation of double SHA-256 hash of the transaction (excluding Transaction ID field).
8 Byte	**Timestamp**	uint64	Unix timestamp: Seconds since 1. January 1970 (UTC) Note: Use 64 Bit to allow dates after 19. January 2038, see http://www.unixtimestamp.com/
4 Byte	**Version**	uint32	The specification version of this transaction. This field could be used to gradually upgrade the network to a new version.
1 Byte	**Type**	uint8	For *ExecutionRequest* 0x00 and for *ExeuctionInstance* 0x01. Extensible for future use.
4 Byte	**ScriptSigLen**	char	The length of the following ScriptSig field in Bytes.
? Byte	**ScriptSig**	char[?]	**TODO:** Figure out suitable scripting language.
4 Byte	**ExtHeaderLength**	uint64	The length of the following extension header. Default: 0. Used for future extensions.
? Byte	**ExtHeader**	char[?]	This field is used for future transaction header extensions and is currently not used (see ExtensionHeaderSize default value zero).
DATA SECTION			
4 Byte	**InputDataRefLen**	char	The length of the following InputDataRef field in Bytes.
? Byte	**InputDataRef**	char[?]	Reference to the input data. Currently recommendation is the use of the magnet URI scheme and as default protocol for file sharing BitTorrent with streaming capabilities.
4 Byte	**ExecutableRefLen**	char	The length of the following ExecutableRef field in Bytes.
? Byte	**ExecutableRef**	char[?]	Same format as Input Data Reference, but points to an executable. This could be a binary, a script or even virtual machine instances that can be executed.
32 Byte	**ResultId**	char	A random identifier that can be used to reference results of this computation. Needed to generate predefined workflows without waiting for results.

TABLE B.2: ExecutionRequest specification with field size and data type.

Appendix C

Binary Block Specification

Size	Description	Data Type	Comments
HEADER			
32 Byte	**MagicNo**	char[32]	Hexadecimal representation of double SHA-256 hash that uniquely identifies the blockchain.
8 Byte	**Timestamp**	uint64	Unix timestamp: Seconds since 1. January 1970 (UTC) Note: Use 64 Bit to allow dates after 19. January 2038, see http://www.unixtimestamp.com/
4 Byte	**Version**	uint32	The protocol version of this block. Could be used to gradually upgrade the network to a new version.
32 Byte	**HashPrevBlock**	char[32]	Hash of the previous block header. Assures that historical blocks, including transactions, can not be altered. Additionally, it creates a chain of blocks.
32 Byte	**HashMerkleRoot**	char[32]	Merkle root hash of all transactions that are part of this block. Allows to remove old transactions from the local store, if they were successfully executed. Resulting in smaller blockchain size, without breaking the verifiability.
TRANSACTIONS SECTION			
4 Byte	**TransactionLen**	char	The length of the following, serialized transaction.
? Byte	**Transaction**	char[?]	Transaction in serialized binary form.

Table C.1: Block specification with transaction list. Last two fields repeat for each transaction.

Bibliography

[And04] David P Anderson. "Boinc: A system for public-resource computing and storage". In: *Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on*. IEEE. 2004, pp. 4–10.

[Bac+14] Adam Back et al. *Enabling blockchain innovations with pegged sidechains*. 2014.

[Bal15] Philip Ball. "The Trouble With Scientists - How one psychologist is tackling human biases in science." In: *Nautilus* (2015). URL: http://nautil.us/issue/24/error/the-trouble-with-scientists.

[Bec+13] Sean Bechhofer et al. "Why linked data is not enough for scientists". In: *Future Generation Computer Systems* 29.2 (2013), pp. 599–611.

[Bra87] Gabriel Bracha. "Asynchronous Byzantine agreement protocols". In: *Information and Computation* 75.2 (1987), pp. 130–143.

[C+99] Miguel Castro, Barbara Liskov, et al. "Practical Byzantine fault tolerance". In: *OSDI*. Vol. 99. 1999, pp. 173–186.

[Cha13] Mark A Changizi. *The brain from 25,000 feet: High level explorations of brain complexity, perception, induction and vagueness*. Vol. 317. Springer Science & Business Media, 2013. URL: https://books.google.nl/books?id=LE3qCAAAQBAJ&printsec=frontcover&hl=nl#v=onepage&q&f=false.

[Cla98] Tim Clark. "Digicash files chapter 11". In: *CNET News. com* (1998).

[FJ10] Ixchel M Faniel and Trond E Jacobsen. "Reusing scientific data: How earthquake engineering researchers assess the reusability of colleagues' data". In: *Computer Supported Cooperative Work (CSCW)* 19.3-4 (2010), pp. 355–375. URL: http://link.springer.com/article/10.1007/s10606-010-9117-8/fulltext.html.

[Ito15] Joichi Ito. *Why Bitcoin is and isn't like the Internet*. 2015. URL: https://www.linkedin.com/pulse/why-bitcoin-isnt-like-internet-joichi-ito.

[Jha+07] Shantenu Jha et al. "Grid Interoperability at the application level using SAGA". In: *e-Science and Grid Computing, IEEE International Conference on*. IEEE. 2007, pp. 584–591.

[Jia+07] Haoliang Jiang et al. "Gstring: A novel approach for efficient search in graph databases". In: *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*. IEEE. 2007, pp. 566–575.

[JMV01] Don Johnson, Alfred Menezes, and Scott Vanstone. "The elliptic curve digital signature algorithm (ECDSA)". In: *International Journal of Information Security* 1.1 (2001), pp. 36–63.

[KKS08] Peter Kacsuk, Tamas Kiss, and Gergely Sipos. "Solving the grid interoperability problem by P-GRADE portal at workflow level". In: *Future Generation Computer Systems* 24.7 (2008), pp. 744–751.

[KSG03] Sepandar D Kamvar, Mario T Schlosser, and Hector Garcia-Molina. "The eigentrust algorithm for reputation management in p2p networks". In: *Proceedings of the 12th international conference on World Wide Web*. ACM. 2003, pp. 640–651.

[Lam05] Leslie Lamport. *Generalized consensus and Paxos*. Tech. rep. Technical Report MSR-TR-2005-33, Microsoft Research, 2005.

[LSP82] Leslie Lamport, Robert Shostak, and Marshall Pease. "The Byzantine generals problem". In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* 4.3 (1982), pp. 382–401. URL: http://people.cs.uchicago.edu/~shanlu/teaching/33100_wi15/papers/byz.pdf.

[Maz] David Mazières. "The Stellar Consensus Protocol: A Federated Model for Internet-level Consensus". In: (). URL: https://www.stellar.org/papers/stellar-consensus-protocol.pdf.

[MN93] Gennady Medvinsky and Clifford Neuman. "NetCash: A design for practical electronic currency on the Internet". In: *Proceedings of the 1st ACM conference on Computer and communications security*. ACM. 1993, pp. 102–106.

[Nak08] Satoshi q Nakamoto. "Bitcoin: A peer-to-peer electronic cash system". In: *Consulted* 1 (2008), p. 2012.

[OO14] Diego Ongaro and John Ousterhout. "In search of an understandable consensus algorithm". In: *Proc. USENIX Annual Technical Conference*. 2014, pp. 305–320.

[Pat80] Richard E Pattis. "Ebnf: A notation to describe syntax". In: *While developing a manuscript for a textbook on the Ada programming language in the late 1980s, I wrote a chapter on EBNF* (1980).

[PHS98] Tomi Poutanen, Heather Hinton, and Michael Stumm. "NetCents: A lightweight protocol for secure micropayments". In: *USENIX Workshop on Electronic Commerce*. 1998, pp. 25–36.

[Poe15] Andrew Poelstra. "On Stake and Consensus". In: *Public Domain* (2015). URL: https://download.wpsoftware.net/bitcoin/pos.pdf.

[Rei85] Rüdiger Reischuk. "A new solution for the Byzantine generals problem". In: *Information and Control* 64.1 (1985), pp. 23–42.

[Roh+07] Kurt Rohloff et al. "An evaluation of triple-store technologies for large data stores". In: *On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops*. Springer. 2007, pp. 1105–1114.

[SKC00] Matthias Schwab, Martin Karrenbach, and Jon Claerbout. "Making scientific computations reproducible". In: *Computing in Science & Engineering* 2.6 (2000), pp. 61–67.

[Str+11] Rudolf Strijkers et al. "Toward Executable Scientific Publications". In: *Procedia Computer Science* 4 (2011), pp. 707–715.

[Swa15a] M. Swan. *Blockchain: Blueprint for a New Economy*. O'Reilly Media, 2015. ISBN: 9781491920473. URL: https://books.google.nl/books?id=RHJmBgAAQBAJ.

[Swa15b] Melanie Swan. "Blockchain Thinking: The Brain as a DAC (Decentralized Autonomous Organization)". In: *Texas Bitcoin Conference*. 2015, pp. 27–29.