How to securely scale the blockchain: a hierarchical child chain protocol with on-chain data auditability and lifecycle management

Moustakas, Nikos

How to securely scale the blockchain: a hierarchical child chain protocol with on-chain data auditability and lifecycle management

Nikos Moustakas

This thesis was written in partial fulfillment of the requirements of the MSc in Cybersecurity from the Cyber Security Academy in The Hague, Netherlands

Colophon

Title	How to securely scale the blockchain: a hierarchical child chain protocol with on-chain data auditability and lifecycle management
Author	Nikos Moustakas
Student number	s2721538
Supervisor	Dr. Zeki Erkin
Second reader	Dr. Kaitai Liang
Institution	Cyber Security Academy
Faculty	Governance and Global Affairs
Program	MSc in Cybersecurity
Date	21 January 2022
Version	Final

Abstract

After the rise of various public blockchains like Bitcoin and Ethereum, there is now a plethora of distributed applications and services based on this paradigm in areas such as finance, security, gaming and voting. However, due to blockchain mechanics this popularity has made several negative effects become obvious, in terms of how many transactions per second can be handled, what are the associated costs per transaction and how can a blockchain grow to fit its ever-expanding capacity requirements of the users. In order to overcome the capacity and scaling challenges, a new blockchain architecture dogma is becoming mainstream, where child-chains or sidechains are connected to the main blockchain as separated entities and can handle a part of the workload. But this in turn gave birth to a new set of security and trust considerations which need to be addressed.

In this thesis we had two goals. The first was to analyze the prominent layer-2 blockchain implementations while focusing on their security built-in mechanisms in order to provide a comparison between them. This comparison was subsequently used to identify areas where a new protocol could provide enhancements. The second goal was to propose a new protocol which we believe addresses the security concerns of the existing solutions. This protocol uses a master-slave architecture, where multiple nested layers are possible in a hierarchical manner. Smart contract functionalities are used in order to create, manage and delete layer-2 blockchains which are derived from a master blockchain. We also show how this protocol can be used by entities to create either a public or a private blockchain to transact and still be able to prove the integrity of the data and transactions on their blockchain. This protocol bears resemblance to the public-key infrastructure where smart contracts on the main blockchain act as the certificate authorities to the layer-2 chain. The smart contracts, used in this proposed protocol, can sign the start of a new layer-2 blockchain, gather signatures of the output in a synchronized or ad-hoc way and, at the end, revoke them flagging them as finished.

As a last step we provided a comparison between the existing solutions and our protocol. The analysis showcased the areas where our protocol provides more auditability and trust to the users. The last part provided the conclusions and further work which needs to be done.

Keywords

Blockchain, Smart Contracts, Security, Layer-2, Scalability, Interoperability.

Contents

Abbreviations

ANSI	American National Standards Institute
BIP	Bitcoin Improvement Proposal
dApp	Decentralized Application
DAG	Directed Acyclic Graph
DAO	Decentralized Autonomous Organization
DeFi	Decentralized Finance
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
EVM	Ethereum Virtual Machine
IEEE	the Institute of Electrical and Electronics Engineers
NIST	National Institute of Standards and Technology
OOP	Object Oriented Programming
PKI	Public Key Infrastructure
PoW	Proof of Work
PoS	Proof of Stake
RSA	Rivest, Shamir, and Adelman algorithm
SegWit	Segregated Witness
SHA-256	Secure Hash Algorithm 256 bits
ZK	Zero Knowledge

Figures

Tables

1. Introduction to the research question

1.1. Blockchain introduction

Bitcoin was born in 2008 in a paper written by an unknown author using the pseudonym Satoshi Nakamoto [1]. This paper described a new electronic transaction-based system which could function without a trusted party acting as intermediate but rather in a peer-to-peer mode, and yet still being able to provide a reliable mean of executing digital transactions for its participants. This novel system used a **public distributed ledger** where all transactions are visible to all blockchain nodes which participate in the network. This contrasted with the traditional centralized or distributed architectures which use trusted nodes for controlling the traffic as shown in the figure below.

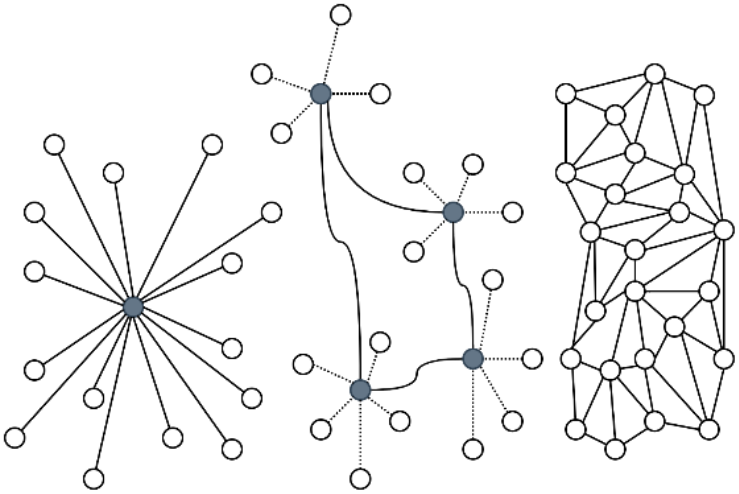

Figure 1: a) Centralized, b) Distributed, c) Decentralized

Moreover, Bitcoin introduced the notions of **immutable transactions** and **consensus mechanisms**. Once the transactions are included in the ledger, after being checked for correctness according to the cryptographic rules of Bitcoin, they cannot be undone or cancelled for any reason without performing an attack to the network. Various implementations which allow the cancellation of transactions while still maintaining the cryptographic chain of signatures have been theoreticized [2] or implemented [3] but have never been adopted by the main blockchain applications due to issues arising with the maintenance of integrity of the ledger [4]. The immutability feature means that the only way to reverse the transaction is for

the recipient to send the bitcoin back to the sender. This is due to the fact that the transactions are included in a block, which in turn is cryptographically tied to the previous block creating thus a chain of blocks; the blockchain. The way to decide how this chain of blocks is built and accepted by all the network participants is the consensus protocol. This mechanism is based on executing cryptographic functions to find a mathematical solution, according to the rules of Bitcoin, which is a concept initially designed to combat spam emails [5]. This process is called mining and the consensus protocol is called **Proof-of-Work** and forms the foundation of Bitcoin's immutability feature. Figure 1 shows a basic diagram where miners construct new blocks, with new transactions from the transaction pool and the cryptographic signature of the previous block.

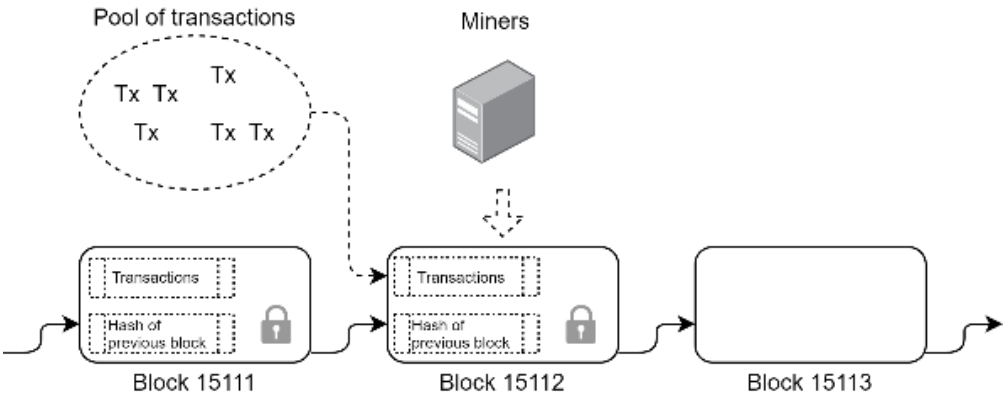

Figure 2: Basic blockchain diagram

Another significant consensus mechanism is the **Proof-of-Stake** where users stake their coins in order to take part in the consensus mechanism, acting as validators [6]. This mechanism does not rely on solving computational problems like in proof-of-work, but rather a validator is chosen randomly which creates the new block. If the validator is a malicious user and tries to attack the network, then the user's stake is lost, and a new validator is chosen.

Following the rise of Bitcoin, other blockchain implementations followed suit. These brought along variations on their consensus mechanisms and the cryptographic functions they use, but they all have the basic characteristics of a blockchain network as set by Bitcoin. In addition, more use cases based on blockchain were introduced, such as voting [7] , using it to provide data integrity for sensitive data [8], or as a register and monitoring system of a food supply chain [9]. Another popular blockchain, **Ethereum,** was introduced in 2013 and brought together a built-in Turing-complete programming language[10]. This meant that in addition to

the exchange of value of implementations such as Bitcoin, Ethereum could be used to develop blockchain run applications, becoming the first programmable blockchain. The latter was achieved with the implementation of the application layer in the form of smart contracts. A **smart contract** is simply software code residing on the Ethereum blockchain and is programmed to execute contract tasks. It is worth noting that the notion of smart contracts defined as software code which can execute the terms of a contract was not introduced first by Ethereum but was proposed for handling financial or accounting transactions in 1994 [11]. This new paradigm led to the development of hundreds of distributed applications which used smart contracts running on a blockchain as part of their operations, with millions of users, in finance, gambling, identity, energy, insurance, property rights and other areas [12]. Figure 2 below showcases a simple diagram on the difference between a smart contract-based application versus a traditionally built internet service.

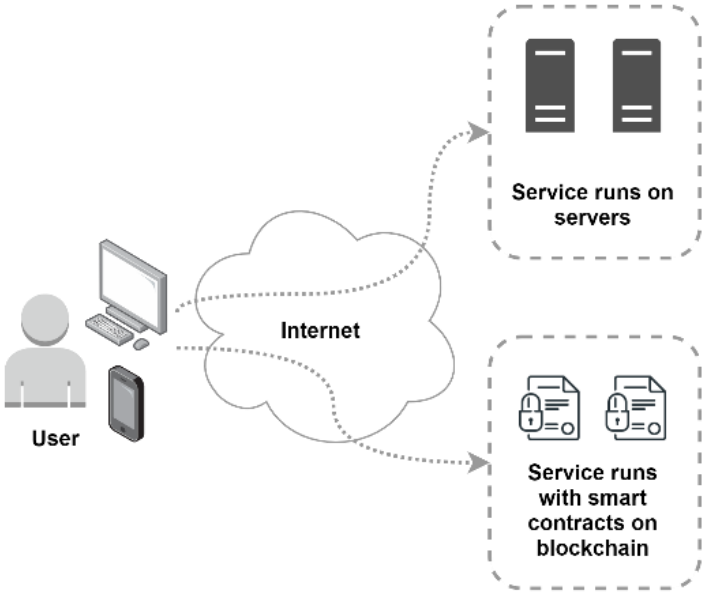

Figure 3: dApp vs traditional architecture

The abovementioned consensus mechanisms and structures of blockchain systems make sure that the transactions are reliable and malicious actors cannot intervene. This however does not mean that there are no **security threats** in other areas of blockchain implementations. Attacks on the network layer include the 51% attack, where an attacker manages to hold enough resources to outpace the rest of the miners in order to manipulate the transaction flow in proof-of-work systems. In this way the attacker can reverse transactions, exclude transactions and block the execution of other transactions [13]. Another area of attack is when multiple malicious actors could take advantage of the propagation delay of newly mined blocks in order

to impact the distribution of mining rewards [14]. In proof-of-stake based implementations, a malicious actor can try to force a chain reorganization by withholding their blocks and releasing them at an opportune time, even if the actor is holding less than the 30% of the total stake [15]. Last, future threats include attacks on the cryptographic functions which underpin the workings of a blockchain system using quantum computers [16].

Another aspect of the blockchain architecture is that it provides less **performance**, in terms of transactions per second, compared to traditional transactional-based networks based on the use of trusted intermediates. As an example, the Bitcoin network can process up to 7 transactions per second and Ethereum up to 20 transactions per second, which is a trivial performance compared to the 24.000 transactions per second which the Visa network can process[17]. This bottleneck created the need for scalability solutions, which can be categorized in layer-1 and layer-2 solutions. Layer-1 solutions include the variations of the transaction model of the blockchain or the size of each block. An example of layer-1 solution is the block size limit of Ethereum, which can fluctuate according to the network needs, from the nominal value of 15 million gas worth of transactions up to 30 million [18]. The layer-2 solutions build side chains which are connected to the main blockchain network but operate in a separate manner. The main variants of layer-2 solutions will be discussed in the next chapter.

1.2. Problem definition

The scalability issue of blockchain is directly related to the '**scalability trilemma**', a concept introduced by the cofounder of Ethereum, Vitalik Buterin [19]. A representation of the trilemma can be seen in the below figure:

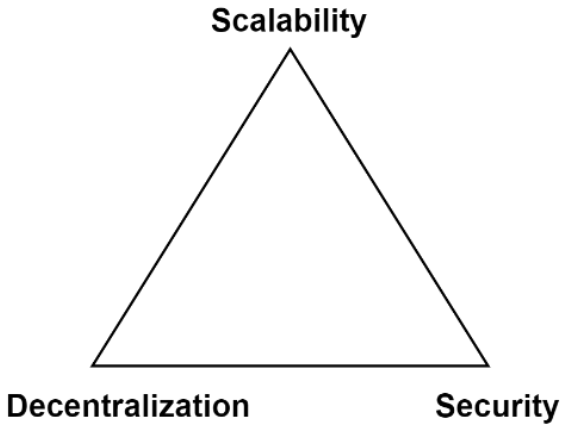

Figure 4: Scalability trilemma

The trilemma states that with basic network structures, an architecture can provide sufficient characteristics for only two out of the three aspects. A network of a bank is highly scalable and can provide sufficient security but there is no decentralization since the bank controls the execution of all the transactions. A file sharing implementation like BitTorrent is both decentralized and scalable but it is not secure [20]. A blockchain based network is decentralized and provides security by means of cryptography and its consensus rules, but by doing so it lacks scalability. Another blockchain network can be permissioned, so only a handful of nodes are eligible to produce new blocks of transactions in a more scalable manner but then the network becomes less decentralized.

When we specifically compare blockchain implementations against the scalability trilemma, both the security and scalability aspects can be further examined according to the functions per item, such as confidentiality, integrity, availability and privacy for security and availability, capacity and latency for scalability [21]. For example, the Binance Smart Chain implementation, which is an EVM-based blockchain [22], uses proof-of-stake-authority as its consensus mechanism. This allows it to produce a new block every 3 seconds [23]. According to the rules of this mechanism, participants stake their cryptocurrency in order to become validators and create new blocks, but critics mention the fact that who becomes a validator is chosen by 11 specific nodes, making thus the network more scalable at the cost of decentralization [24]. This is because these 11 nodes have more administrative control over the execution of transactions.

1.3. Motivation and contributions

The first goal of this thesis is to identify the current layer-2 blockchain architecture types and catalogue a few of the most prominent layer-2 commercial implementations. We also provide an explanation on how they address the scalability challenge. We focus on their security features and construct metrics which we use to compare them.

Moreover, our second goal is to propose our own protocol which we believe addresses the scalability issue and has security features regarding integrity. This protocol is based on our prior work which was registered in the form of two patents[25], [26]. This work provides the foundation on which our proposed protocol is designed. In this protocol we propose a novel

blockchain architecture which aims not only to provide scalability and security, but also auditability. The term auditability refers in this case in the ability of a blockchain user to be able to check and trust all relevant transactions made not only in the main blockchain but also in the child chains. The later includes the possibility where the child chains are private blockchains with their own consensus mechanisms and transaction schemes. We believe that auditability enhancements can further increase the trust of end users to use blockchain based products and services.

1.4. Outline of the thesis

We decided to use the first part of the thesis to provide technical information about foundational topics which the reader needs to comprehend. This knowledge is needed for better understanding the concepts of the next chapters. These topics include the basic innerworkings of a blockchain system, how a blockchain transaction is built and what information is used, how the cryptographic functions protect the security of the identities and the network and more. Additionally, we describe the application layer of a blockchain network which includes the smart contracts. We provide information about how smart contracts work and how distributed applications, in short dApps, are using them as part of their backend systems.

The second part of the thesis contains a survey of current layer-2 blockchain implementations. In there we analyze the different categories of this solutions, how they address the scalability challenge, and finally their built-in security features. We focus on the user viewpoint and investigate if and how a malicious actor can utilize layer-2 attacks.

The third part introduces the new protocol. We go in depth following the analytical framework of the previous chapter so that we can later make comparisons with current solutions and draw conclusions. We showcase the basic features of this protocol and enumerate the security implementations regarding malicious attacks to the integrity of the system. We also provide information on how this protocol provides additional trust to the users.

The comparisons between the current solutions and the proposed protocol are included in the fourth part. The target of this thesis is to identify at least one metric where our protocol provides superior security. In the fifth and last part we summarize the conclusions and provide insights and suggestions on further work.

1.5. Naming conventions

Please note that the terms *technical layer*, *socio-technical layer* and *governance layer* are used in the manner shown in figure 4 [27]. The technical layer can be subdivided into network and application layer. The network layer includes the basic blockchain architecture and tools whereas the application layer includes the smart contracts and distributed applications. Furthermore, the term *layer-1* is used to describe scalability solutions within the same blockchain implementation. The term *layer-2* is used to describe scalability solutions which are running outside and parallel to the main blockchain network and should not be confused with the application layer.

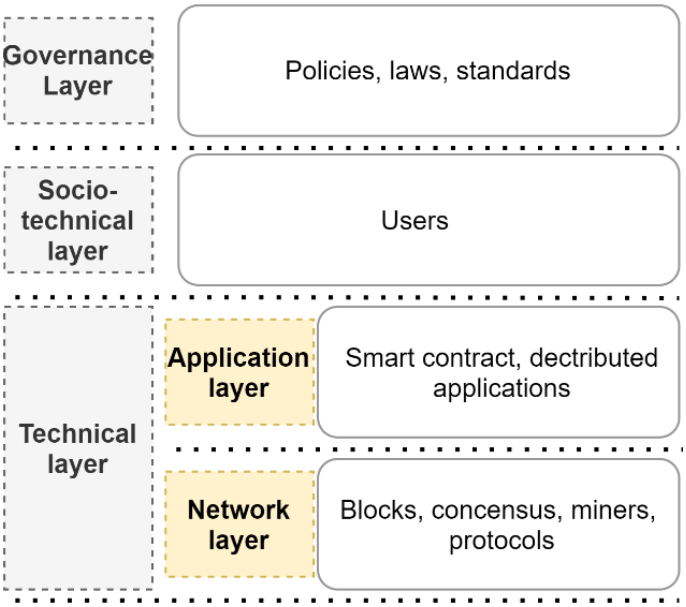

Figure 5: Naming conventions

2. Background information

For the background information we will mostly use Bitcoin and Ethereum as basis for the explanations since these are the most prominent blockchain implementations, totaling 70% of the total blockchain market [28].

2.1. Blockchain cryptography

A blockchain network is using cryptographic primitives as the basis of its operation, and more specifically, assymetric cryptography, digital signatures and hash functions. **Assymetric**, or public key, cryptography is used to produce a public key from a randomly chosen private key, and to thus create the pair of keys which every use account rely upon. This process is a one-way process which means that an attacker cannot calculate the private key just by knowing the public key. This set of private and public keys are used for every user account in blockchain, for signing transactions. For example both Bitcoin and Ethereum use *Eliptic Curve Cryptography (ECC)* and more specifically the secp256k1standard [29] [30], where the generation of a public key P is calculated by multiplying the private key K by a predetermined point of the elyptic curve, called generator G:

$$P = K * G,$$

where the generator G is defined in the secp256k standard as G = 0279BE667E F9DCBBAC55A06295CE870B07029BFCDB 2DCE28D9 59F2815B 16F81798 [31]. Even if the same generator G is used for the generation of the public key, there is no security threat since the calculation happens only in one direction.

Hash functions are one-way funtions where an input of arbitrary size is transformed into fix length character data. An important chatacteristic of such fuctions is the non-correlation of the input and transformed data, called the avalange effect, where even the change of 1 bit of the input produces a significant change in the transformed data [32]. Hash functions are also deterministic, meaning the same input always calculates to the same result, are computational easy to calculate, and there are collision resistant which means that different inputs should not produce the same result. Bitcoin uses the *Secure Hash Algorithm (SHA-256)* hash function,

while Ethereum uses the *Keccak-256* hash function [30], [33]. Both of these functions produce an output of 256 bits, but utilizing multiple rounds of bit manipulation and permutation. Hash functions are used in blockchain to create a public address from the public key, 'chain' the blocks with each other, for the proof of work consensus mechanism and more.

Digital signatures are functions used to sign digital data in a similar way a physical signature is used to sign physical documents. They are mathematical functions where the owner of the private key encrypts the hashed output of a data object, which is called the digital signature, and the receiver decrypts the data using the owner's public key and compares the data to verify that the signature is valid. Digital signatures are used in blockchain for *authorization*, *non-repudiation* and *data integrity.* The authorization is proven since only the owner of the private key can produce a valid signature, which thereafter can be verified by anyone who has the data and the public key of the owner. Since only the owner of the private key can produce the signature, it means that no one else could have produced this signature, providing thus non-repudiation. Finally, the signature provides data integrity because a different set of input data would produce a different digital signature when signed with the same private key. Both Bitcoin and Ethereum use the *Elliptic Curve Digital Signature Algorithm (ECDSA*) which is a standard accepted by ANSI, IEEE and NIST [34] and is based on the usage of the eliptic curve private and public keys. ECSDSA can provide the same level of security with smaller key size (256) compared to other algorithms like RSA (2048) [35].

2.2. Blockchain basics (network layer)

A **blockchain address** is derived from the public key of a user. Bitcoin creates the address by first hashing the public key with the sha-256 function and then hashing again the outcome with the Ripemd160 function [33]. The last step is to encode the result with the Base58check scheme, producing a 58 character public address.

A blockchain **transaction** is a transfer of value from one account to the next, utilizing the cryptographic functions mentioned in the previous paragraphs. The owner of an unspent value, like a bitcoin, which was previously sgined to his public address, can now use his private key to produce a signature to verify his identity. In parallel the value is now encrypted with the public key of the new receipiant.

A **block** contains all the information needed to add the latest transactions in the ledger. It has a header with important information, such as the has of the previous block, a timestamp, a nonce in the case of proof of work consensus mechanism and the merkle root hash of the transactions included in this block. The main body of the block then contains the actual transactions [36]. The table below showcases the content of the header of a Bitcoin block:

Table 1: Bitcoin block header contents

Size	Field	Description
4 bytes	Version	Version number
32 bytes	Previous block hash	Reference to previous block
32 bytes	Merkle root	Root hash of Merkle tree of transactions
4 bytes	Timestamp	Block's creation time
4 bytes	Difficulty target	Difficulty of proof of work algorithm
4 bytes	Nonce	The solution to the proof of work problem

An **oracle** is a data feed which serve smart contracts by providing off-chain inputs and outputs. For example, a prediction market smart contract needs input for the results of an event in order to settle bets [37] or to ensure the bidding starting and end times of an auction [38]. There are various types of oracles like hardware ones delivering sensor data or software ones gathering information from web applications. Oracles can input data into the blockchain but can output events as well.

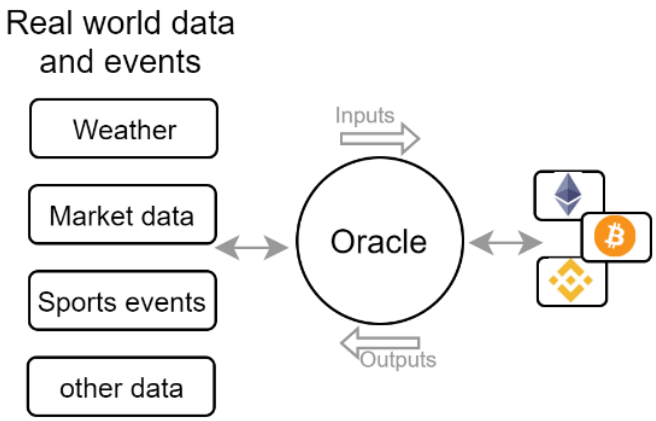

Figure 6: Oracles and blockchain

Consensus mechanisms are protocols which provide fault-tolerant transaction verification. We have already discussed about Proof of Work and Proof of Stake, but there are many more

protocols such as Proof of Working State, Proof of Stake Time, Proof of Work Time, Proof of Space Time, Proof of Devotion, Proof of Activity, Proof of Luck, Proof of Authority, Proof of Credit, Proof of Location, Proof of Credibility, Voting and more [39]. Every consensus mechanism scores differently in areas such as scalability, finality, accessibility and adversary tolerance.

A distinction of blockchains between p**rivate and public blockchains** is their accessibility. Both Bitcoin and Ethereum are public and permissionless blockchains, meaning that their ledger is visible to every node which participates in the network and every node is allowed to participate and generate new blocks. On the other hand, a private blockchain of permissioned is one where an authority can control who can participate in the network and view the ledger. Examples of such private blockchain is Ripple [40] and Hyperledger Fabric [41]. Both private and public implementations have variations in their performance[42] or their auditability [43].

2.3. Smart contracts and decentralized applications (application layer)

EVM, which stands for **Ethereum Virtual Machine**, is the runtime environment for smart contracts in Ethereum. This makes Ethereum a 'Turing complete' state machine where every new state is compiled by the previous state and the new transactions[44]:

$$\sigma_{(t+1)} = Y(\sigma_{(t)}, T)$$

, where σ is the state, Y is the execution of the EVM state transition function and T are the new valid transactions included in the latest block. The whole state of Ethereum is contained in one data structure called the Merkle Patricia Trie, which is a combination of a Merkle tree and a Patricia Trie [45]. The figure below shows a graphical representation of the EVM.

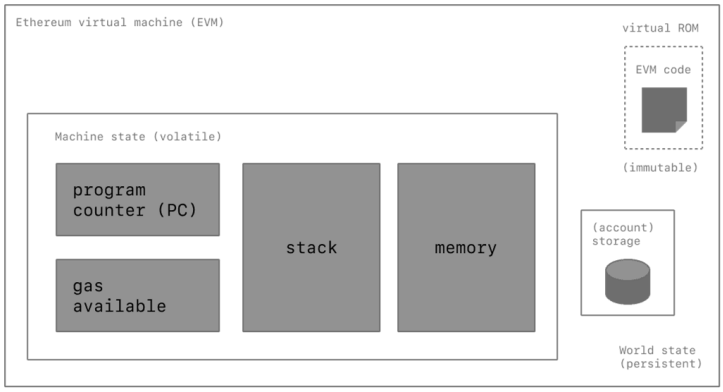

Figure 7: The Ethereum Virtual Machine [44]

Smart contracts are programs which run on the blockchain. In Ethereum every deployed smart contract has its own account address and own balance. They can receive input in the form of transactions, execute their code and interact with other addresses. Once deployed on the blockchain a smart contract cannot be deleted or modified since the immutability feature apply to them as well, which poses also a security threat which will be analyzed in the last section of this chapter. The interactions with a smart contract are also irreversible since they modify the state of the blockchain from one block to the next [46]. The smart contracts are uploaded on the blockchain in the form of bytecodes which use several opcodes for their functionality. In Ethereum the opcodes perform stack, memory and storage manipulation, arithmetic functions and halting operations [47]. With these opcodes and enough gas, a smart contract can be developed to execute any computing logic or function. A simple smart contract which has only two functions, one which stores a value, *store()*, and one which retrieves it, *retrieve()*, can be seen below. This code snippet is one of the examples from the integrated development environment used for smart contract development, called Remix [48].

Home | 1_Storage.sol

```
1  pragma solidity >=0.7.0 <0.9.0;
2
3  contract Storage {
4
5      uint256 number;
6
7      function store(uint256 num) public {
8          number = num;
9      }
10
11     function retrieve() public view returns (uint256){
12         return number;
13     }
14 }
```

Figure 8: Storage smart contract

The above example is written in **Solidity**, which is the defacto development language of Ethereum [49], while other languages include Vyper, Yul and Yul+ [50]. Solidity is a statically typed, case-sensitive and object-oriented programming (OOP) type language and comes with its own compiler, called 'solc', which compiles the solidity code into the bytecode which runs on the EVM of Ethereum[51]. This bytecode is run every time there is a transaction which calls one of the functions of the smart contract. In the example of figure 6, any transaction calling the function *store()*, with a numeric value as input, would cause the EVM to run the specific bytecode and store the number as a modification of the Ethereum state assigned to the account of this contract.

Decentralized applications are applications which run on a blockchain and use smart contracts for their functionality. They are open source, and the code logic is operated autonomously within the smart contracts. They are deterministic and since they run on a blockchain, they have no down time, their data integrity is secured, and they have a verifiable behavior [52].

2.4. Layer-1 scalability solutions

As we mentioned in the introduction, the focus of the thesis is the layer-2 scalability implementations. These are implementations where another system is built next to the main blockchain with the aim of taking over a percentage of the workload. In contrast to that, layer-1 implementations are targeting the blockchain network itself by changing some of its characteristics. Below we mention a few of layer-1 scalability solutions.

A first characteristic which would provide scalability would be to increase the number of transactions included per block, meaning increasing the size of the block. For example, the current size of a block in Bitcoin is 1 MB, as explicitly written in its codebase [53]. Currently, the statistics show that the average number of transactions in 1 MB block is 1477, reaching a maximum of 2768 [54]. There have been various feature requests, the so-called Bitcoin Improvement Proposals [55], addressing this issue and requesting blocks of bigger size. More specifically, BIP 101 proposed doubling the block size every 2 years, reaching a maximum size of 8 GB [56]. BIP 103 proposed the increase of the block size by 17,7% per year [57]. Both BIPs have been rejected by the bitcoin community. Other BIPs have been adopted by producing

hard forks of Bitcoin. An example of this is BIP 109 which proposed a block size of 2 MB [58] and was implemented in the form of a hard fork, called Bitcoin Classic [59].

Changes in the consensus mechanisms could also help with scalability. Examples are Bitcoin -NG where every node which wins the proof of work race can produce multiple blocks in the form of a main block and many micro blocks [60] or Byzantine fault-tolerant based blockchains, which sacrifice the total amount of allowed nodes for scalability [61].

Another layer-1 scaling solution is **sharding**. This is considered a horizontal scale expansion where the ledger is split into multiple shards and nodes are allowed to participate in a few shards. This solution was actually inspired by a scalability option in databases where the whole database is partitioned into smaller data segments and stored in servers in multiple locations [62]. There are different sharding implementations with variations about the cross-sharding transaction processing, confirmations, use of committees and more [63]. Ethereum is implementing sharding in the upcoming upgrades which will lead to the Ethereum 2.0 phase [64]. According to the Ethereum implementation, a few proof-of-stake validators which are randomly assigned will be responsible for creating shard blocks in a fixed interval [65]. RapidChain is another example which uses an intra-committee synchronous consensus protocol [66].

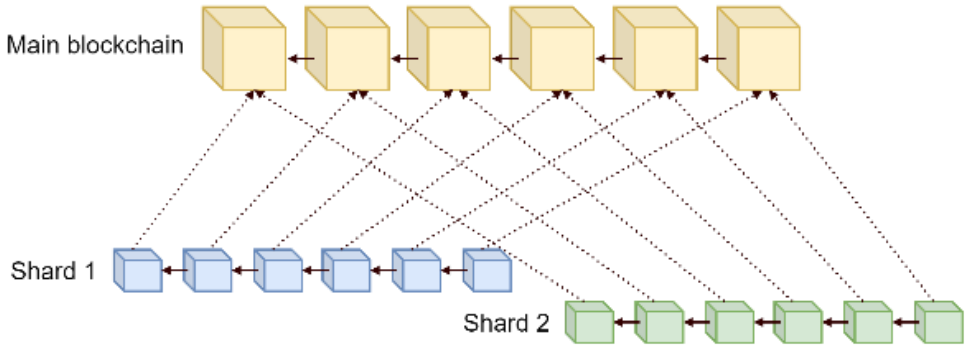

Figure 9: Blockchain sharding

A fourth option is the use of **Directed Acyclic Graph based blockchains**. In this type of blockchains instead of a chain of blocks, there is a chain of transactions in the form of a tree, where one transaction branches from another [67]. There are many implementations of DAG-based blockchains such as IOTA [68], Spectre [69] and Nano [70].

2.5. Security challenges of blockchain implementations

In the network layer we identify multiple vulnerabilities. First, the platform themselves have software errors which lead to vulnerabilities which can be exploited by attackers [71], [72]. Furthermore, when we investigate the consensus mechanisms and even with the scale which some blockchain implementations have reached, there is still the threat of executing denial-of-service attacks in proof-of-work based systems, by targeting their incentives in order to enforce miners to stop mining [73]. Finally, and looking at the future, the cryptographic functions which underpin the security of all blockchain implementations, will be vulnerable to quantum-based attacks [16]. The last vulnerability mentioned in the network layer is the interaction between the digital and physical world using oracles. Smart contracts designed to make decisions based on external sources can be manipulated if an attacker can successfully attack this source and alter the input data. An example of this is the price manipulation of the price of a cryptocurrency in a specific exchange which was monitored by oracles serving DeFi decentralized applications, which caused a liquidation of 100 million USD [74].

In the application layer, we observe the vulnerabilities and threats against smart contract and decentralized applications. Smart contracts are susceptible to various attack techniques such as reentrancy, smart contract overflow, short address attack, delegatecall, default visibilities, transaction ordering dependance, and timestamp dependance [75]. Especially the reentrancy attack is considered to be one of the most severe ones, and it happens when two smart contracts interact with each other and one of them gaining control over the other [76]. A last perspective to the security of smart contracts is the challenge of lifecycle management related to immutability. Relative research has examined the way that software development of such applications has to change in order to accommodate the characteristics of the blockchain [77]. Another issue related to immutability of smart contracts and their open-source nature is that once the bytecode of a smart contract is deployed on the blockchain, it is available to attackers to analyze and search for security threats [78]. Solidity has also security vulnerabilities, such as buffer overflows, exception handling, generation of random numbers and private definitions of variables [79]

Moreover, the popularity of decentralized applications, which rely on smart contracts, has created a new attack surface for malicious actors. Only in 2020 there have been 31 exploited decentralized finance projects, DeFi, with a total loss of capital of 315,95 million USD [80].

One of the first and famous attack on the DeFi space of the application layer was the DAO attack which happened in 2016 [81]. The DAO attack happened when an attacker exploited a reentrancy vulnerability of the DAO smart contract and was able to withdraw DAO's funds to a malicious contract. After a vote of the users of the Ethereum community it was decided to proceed with a hard fork in block number 1.920.000 [82]. With this fork, the stolen funds were transferred to a new contract where the DAO tokens' owners were able to withdraw their funds [83]. A portion of the users opposed the fork because the hack was not considered to be an actual fault of the Ethereum protocol, and continued with the original chain, forming the Ethereum Classic [84].

3. Survey of current layer-2 blockchain scalability solutions

In this chapter we are going to investigate the current layer-2 solutions which aim to increase scalability by executing transactions outside the main blockchain, while maintain the decentralization aspect of the scalability trilemma. There are many implementations available, but they all fall under three main categories: state channels, sidechains and roll-ups. These categories have different characteristics which we are going to analyze, in terms of security and scalability.

3.1. State channels

State channels allow the execution of transactions between users outside the main chain. This is done by initially locking the state of the participating users using a multi-signature scheme or a smart contract. This means that the locked funds are not allowed to be unlocked unless all the participating users agree to do so. After that, the users can sign and execute transactions and update the state, in a peer-to-peer manner without transmitting these transactions to the main chain. Every new transaction updates the state and at the end the users simply publish the last state to the main chain. Then the state in the main chain is again unlocked and the state channel is considered closed.

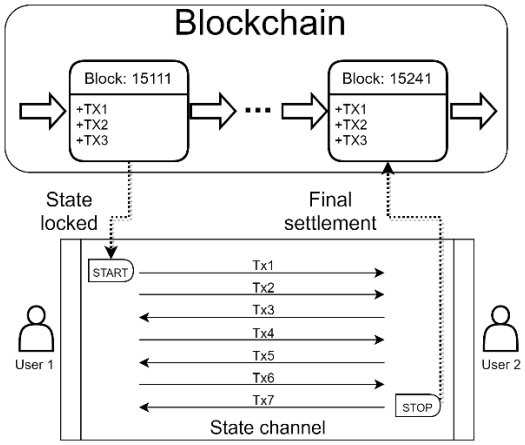

Figure 10: State channel illustration

In store of value based blockchain implementations such as Bitcoin, the state channels are used as payment channels between participants with no or lower fees. In programmable blockchains

such as Ethereum, the state channels can be used to transact generic state updates. An example of this are the moves of a blockchain-based chess game, which is run in a state channel with no fees except the last transaction which will announce the winner in the main blockchain [85].

Scaling is achieved by allowing the transactions to happen off chain, without having to use resources of the main blockchain to execute them. Once the channel is established, the blockchain does not need to process or verify the transactions according to the consensus protocol. Moreover, since no resources are needed, there are also no fees used, making this scaling method a suitable solution for micropayments, or for multiple transactions between two participants. A disadvantage of state channels is that they have to be ad hoc opened and closed per request, and for a specific set of users since when a channel is started no other users can participate. This means that they are not permanent or open for participation in nature.

One security aspect of state channels is the dispute mechanism. This is needed in case a malicious user publishes, not the latest state back to the blockchain, but a modified one or not the latest and tries to close the channel. The dispute mechanism allows for the other participants to publish their latest state so that the proper one is actually handled in the blockchain. This dispute mechanism also carries a penalty for the malicious user.

Even if the transactions are not published in the blockchain, they still need to be signed according to the rules of each blockchain. This mitigates the risk of having a malicious user trying to alter the updated state of the channel.

State channels also provide privacy. This is because the transactions are not handled in the public ledger of the blockchain but between the users participating in this channel. The only visible items in the blockchain are the first and last state of the channel, not the transactions which were handled in the state channel. Also visible to the blockchain are the users which participate in the channel.

For a state channel to remain active, all parties need to be available. If a user becomes unresponsive then it could mean that the other user can close the channel posing thus a security risk. This ‘liveness’ requirement could be mitigated by delegating their presence to other services.

An example of a state channel implementation is the Lightning Network [86]. It offers a solution of executing millions of transactions per second, with low fees. Lightning network

also supports cross-blockchain functionality where users of the network can perform transactions between different blockchains.

3.2. Sidechains and child-chains

Sidechains were introduced in 2014 as a mean to enhance scalability and interoperability by enabling the transfer of assets between different blockchains [87]. A sidechain is constructed as a blockchain which uses a two-way peg mechanism in order to lock assets in the main blockchain and transfer them to the sidechain. Subsequent transactions can happen in the sidechain and at the end the assets can be transferred back to the main blockchain. Assets are locked by using a Simplified Payment Verification method, or SPV, which functions as the proof of possession. A contest period is also used during this peg in order to prevent double spending.

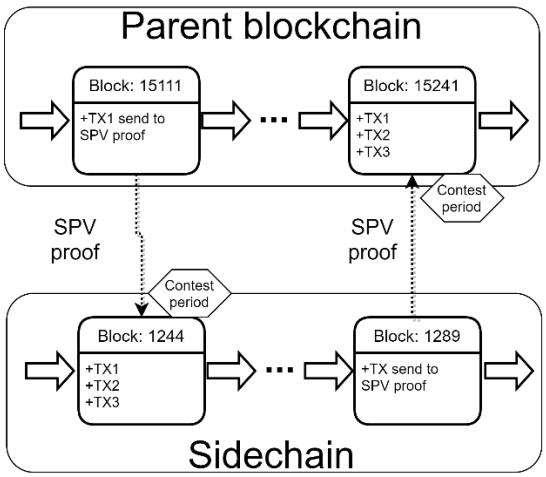

Figure 11: Sidechain illustration

Scalability is achieved by allowing the process of the transactions to happen in the pegged sidechain, without using the resources of the main blockchain. This sidechain can be designed with its own characteristics in terms of the rate of new blocks generation, the consensus mechanism, or the fee structure. The sidechain can also help with offloading specific type of transactions, for example contacting an election vote in a sidechain rather than in the main chain [88].

The main security consideration is the fact that the sidechain operates with its own set of characteristics and the security features of the main blockchain are not imported to the sidechain as well. This could potentially mean that the nodes can be more susceptible to malicious behavior, or that simply the chain can stop producing new blocks, not allowing any user to withdraw their assets back to the main blockchain. Literature shows that sidechain specific consensus protocols have been proposed which attempt to enhance this security aspect within a sidechain without sacrificing the scalability aspect. An example of that is Cumulus [89]. Another mitigation technique is the use of a federated two-way peg system where a federated group of entities function as the custodians of the locked funds and regulate the transfer of the assets between the main chain and the sidechain in case of an attack [90].

An example of a specific type of sidechain with additional security features is the Plasma chain introduced in 2016 [91]. The plasma chain is considered to be a child chain and not a sidechain, since it is connected to the main Ethereum blockchain and uses its security mechanisms because it relies on smart contracts which are executed on the Ethereum. Plasma also functions in a non-custodial way which means that the users can transfer their assets back to the main chain even if the plasma chain goes offline.

3.3. Roll-ups

The third layer-2 option which we are going to examine are rollups. This mechanism is considered the main component for Ethereum scalability, according to the roadmap [92]. With rollups the execution of the transactions is moved off chain, but the data relevant to these transactions are stored in the main chain. This happens with the use of a smart contract deployed in the main chain, which holds the state root, or another compressed data representation of it. With every new batch of transactions which are executed off chain, there is an update of the state root in the main chain. Since the state is stored in the main chain, it means that any node can reproduce the transactions which happened off chain, if they wish to do so in order to detect frauds.

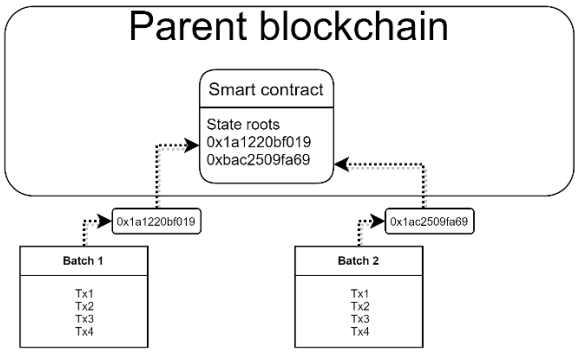

Figure 12: Rollups illustration

There are two main categories of rollups, optimistic and zero knowledge, or ZK, rollups. Optimistic rollups use fraud proofs, by keeping the complete history of state roots along with the hash of every new transaction batch. This allows a user to dispute a specific state root by publishing a proof in the main chain. If this proof is correct, then the smart contract reverts to the correct state root. ZK rollups, on the other hand, use validity proofs, where every new transaction batch must include a ZK-Snark cryptographic proof. This serves as prove that the update state is indeed correct.

Scalability is improved by moving the computation off chain and by lowering the fees needed for the execution of the batches, since only the updates of the state roots use resources of the main chain. Moreover, the cryptographic proof of the ZK rollups provide additional scalability since it uses only the data needed to provide correctness of the batch and no additional state roots need to be processed and stored in the main chain for disputes. With this mechanism, it is possible to fit 62.500 transactions in a single Ethereum block which translates into 4800 transactions per second [93].

An example of optimistic rollups implementation is Optimism [94]. This uses a smart contract deployed on Ethereum, and specific nodes called sequencers which are responsible to execute the transactions of the rollups and report back to the smart contract. If a discrepancy is found during the dispute period, the rollup generates a fraud proof which is executed in the main chain.

3.4. Comparison between different options and identification of security concerns

From the different layer-2 options which we discussed in this chapter, we drew some conclusions about their scalability and security features. The main aspects which need to be investigated are the inherited or standalone security features, the time needed and the functionality for performing the disputes and the effect that all of these have to scalability. The table below summarizes theses insights which will be used in the final chapter for comparison purposes. Moreover, we can make the argument that there can be a separate analysis of every layer-1 and layer-2 solutions against the scalability trilemma as described in the first chapter. With the last column, we try to visualize the impact that every solution would have on a fictitious blockchain which scores a perfect central position in the trilemma. Every plus symbol means that one of the aspects is improved and the minus means that they are weakened.

Table 2: Baseline comparison table

Name	**Layer**	**Main functionality**	**Scaling method**	**Security aspects**	**Example implementations**	**Comments**	**Scalability trilemma**
Block size adjustment	1	Increase the size of the block.	More transactions per block. Also, with the Segregated Witness Protocol, the space for transactions is increased by removing signature data [95].	The same security mechanisms as the main chain.	Bitcoin Classic, SegWit	The nodes spend more resources for validating the bigger blocks.	Scalability: + Decentralization: - Security: Equal
Consensus protocol modification	1	Modify the consensus protocol.	Changing the consensus parameters in order to produce new blocks faster.	Depending on the implementation, this solution can impact the security or decentralization characteristics.	Voting, PoSA	Can be arbitrarily chosen by the designer.	Scalability: + Decentralization: - Security: -
Sharding	1	Ledger is split into shards.	Transactions happen in parallel in each shard. Scaling is increased with every new shard.	Corruption of the nodes in a shard can lead to permanent loss of the data of this shard [96].	RapidChain	Ethereum will enable shards in 2022.	Scalability: + Decentralization: - Security: -
State channels	2	Use of peer-to-peer channels outside the main chain for transactions between parties.	Within this channel, there can be multiple transactions executed, without using capacity of the main chain.	Enhanced privacy 'Liveliness' requirement of the participants.	Celer Network, Lightning Network, Trinity, Raiden Network	Users of the channel must remain online. They exist temporarily.	Scalability: + Decentralization: Equal Security: + -
Sidechains	2	Side or child chains pegged to main chain.	The sidechain can be designed according to throughput needs.	Security is not ensured by the main chain.	Plasma, Cumulus.	Permanent in nature.	Scalability: + Decentralization: Equal Security: -/+
Roll-ups	2	Computation of transactions is moved off chain, while data remain on chain.	Transactions are executed off chain and only the state root is published in the main chain. Compression of the data used also enhance scalability.	Data is kept in the main chain, making it easy to audit the execution of the transactions.	Loopring, zkTube, Aztec, Starkware, zkSync, Optimism, Arbitrum		Scalability: + Decentralization: - Security: Equal

In the above table, we observe that the listed layer-2 solutions provide indeed scalability enhancements but in the expense of security and trust. Sidechains rely on their own security characteristics, except for Plasma based solutions. State channels are temporary in nature and thus we believe cannot be a suitable solution for a blockchain based application. Roll-ups provide data auditability with the mechanism of updating and storing the state roots in the main blockchain, but still fail to provide trust about the operating nodes which create the virtual machines of the layer-2 rollup. These considerations could prevent companies to use these solutions and rely more on private and permissioned blockchains [97]. But by doing so, it creates a data stream which is not public and therefore cannot be easily trusted by the end users. In supply chain cases, auditability and traceability of the data is of outmost importance[98]. An end user who wants to trace the origin of a received product cannot trust a private based blockchain which is managed by the supplier of the product.

4. New protocol description

In this chapter we are introducing the new protocol and the possible benefits it brings. We also name the basic features which showcase the functionality of the design.

4.1. Description

In the previous chapter we analyzed the prominent layer-2 solutions. We have also identified the problems related to security and trust for companies providing blockchain based products and services, and users who consume them. Our protocol is based on the same principles of Public Key Infrastructure, where there is a certificate chain from trusted authorities which issue and maintain digital certificates to issuers, and then issuers in turn issue and maintain digital certificates to end users[99]. This creates a trusted chain since an end user can trace the digital certificates of a website up to the root level. We believe that the same kind of trust can be built in a layer-2 scalability solution.

Our protocol proposes a hierarchical architecture where a public blockchain can be used as the trusted authority. On this blockchain a smart contract controls the issuance and revocation of layer-2 pegged child chains. The figure below showcases the architecture:

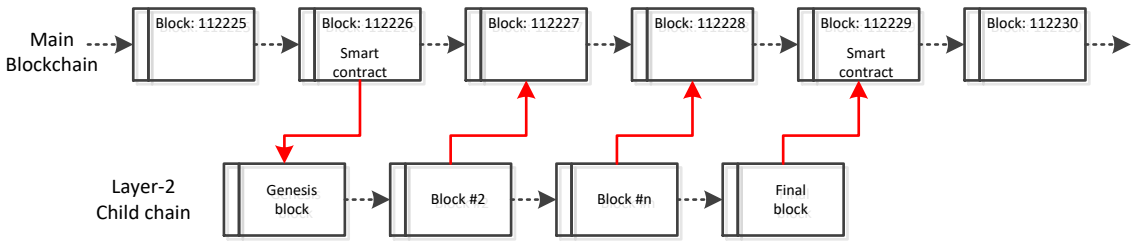

Figure 13: High level architecture

If we reference the hierarchical model used in the Public Key Infrastructure, we assume a similar role of the Root Certificate Authority to the main blockchain. From this layer, new child chains can be created. This mechanism introduces the concept of trusted lifecycle management for child chains. These layer-2 chain can be created, audited and deleted and all this data is available on the main blockchain.

Moreover, we introduce multilayer support. A level 1 child chain can in turn create level 2 child chains which are pegged to them. This creates the following architecture:

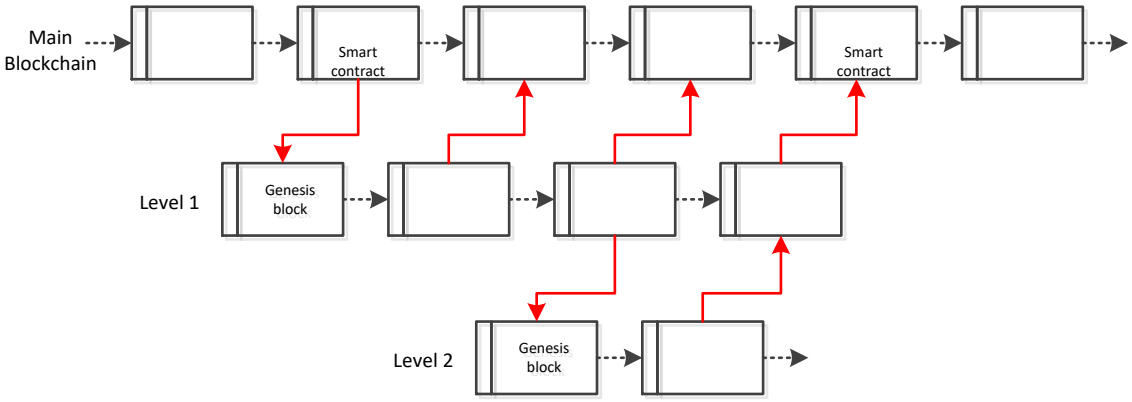

Figure 14: Hierarchical multilayer approach

Every child chain is responsible for the lifecycle management of the child chains pegged to them. It should be noted that due to the auditability of the lifecycle management, the child chains can also be private blockchains without minimizing the trust concerns which we mentioned in the previous chapter.

Finally, our protocol mandates the update of the root state in the main blockchain with every new block of the child chain. This is similar functionality to rollups and accommodate the data audibility characteristic. This provides the data security feature also in the case where the child chain is a private blockchain implementation.

4.2. Creation and validation of a child chain

With this application a child chain can be created via the main blockchain. To accomplish this a transaction executes the smart contract which is stored in the main blockchain. This smart contract assigns a unique identifier and order the creation of the genesis block of the child chain. For further auditability purposes, the identifier could be related to an actual digital certificate assigned to a company which would run this child chain. The unique identifier can also be a signed transaction to the public address of the layer-2 operator, or a multi signature in the case of multiple operators.

The child chain is then activated by means of creating its genesis block. The data contained in the genesis block can include the unique identifier and other characteristics such as the consensus mechanism which will be used, and a timer which signals the heartbeat of the creation new blocks and state updates.

The whole genesis block is then hashed, and the hash code is included in the main blockchain as transaction which updates the state in the smart contract. The following diagram shows the functionality:

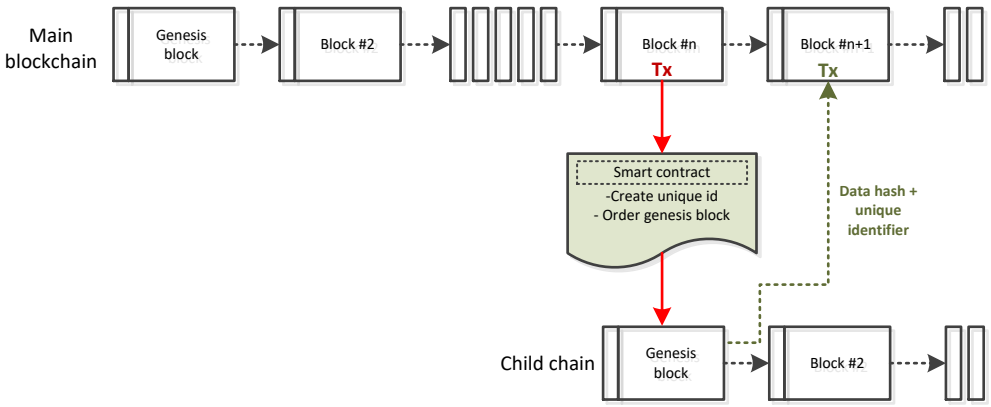

Figure 15: Creation of child chain

The timer mentioned above facilitates the 'liveliness' characteristic of the child chain. In all examples above the creation of new blocks for the main blockchain and child chain is depicted as synchronized. But there might be implementations where this behavior cannot take place. For example, a child chain may choose to create a new block every 60 minutes whereas the root blockchain creates a new one every 10 minutes. This could potentially impose a problem since there would be no way to actually know by interrogating the root blockchain if the secondary blockchain is in faulty status or not. But with the addition of the timer the child chain would simply broadcast the time after which a new block would be created, and the root state would be updated. The new block could also be created without any transactions only to honor the agreement. If the main blockchain does not receive any new information from the secondary blockchain in the predefined time, then the status of the child chain would be set to suspended.

A temporary unique identifier may also be created every time a new block is created in order to be compatible with the sequential principle:

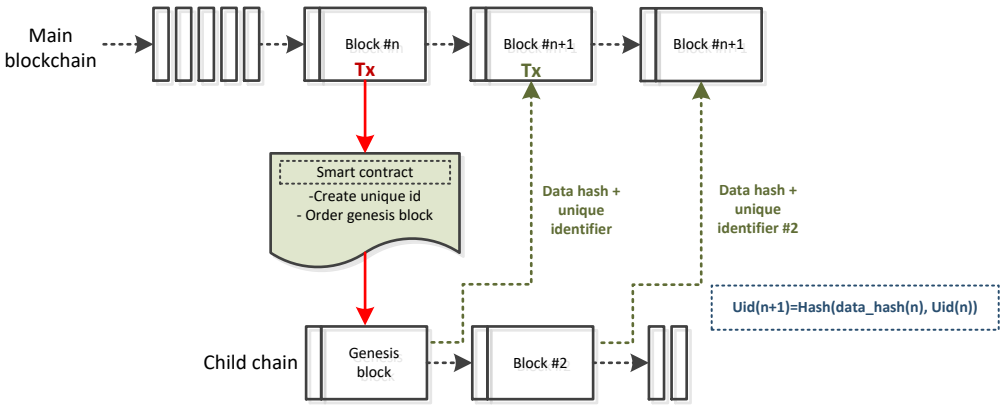

Figure 16: New block creation timer

4.3. Revocation of a child chain

As explained in the previous paragraph, a child chain can be created and maintained by the main blockchain. The last part of the lifecycle management process is how to revoke the child chain. For this the smart contract stored in the main blockchain controls the deletion of the child chain. The output of the contract is the revocation of the unique identifier. The diagram below shows the functionality:

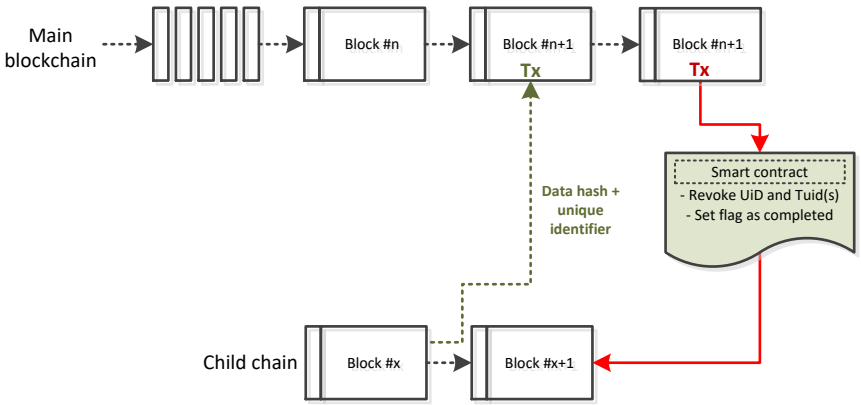

Figure 17: Revocation of a child chain

The revocation in this case can be issued as a new transaction in the memory pool of the child chain. When the nodes of the child chain receive this transaction, then they would consider this child chain, produce the last block and update the state root in the main blockchain. Another mechanism would be to set the status of the child chain as revoked and simply dismiss any new state update.

In order to ensure that a child chain would not execute a new transaction after the revocation procedure has happened, a database of all valid identifiers and temporary identifiers is proposed to be created and as part of the state of the main blockchain. During the validation procedure of the new transactions, if a node finds an expired identifier being used then transaction will be discarded.

4.4. Use case: supply chain

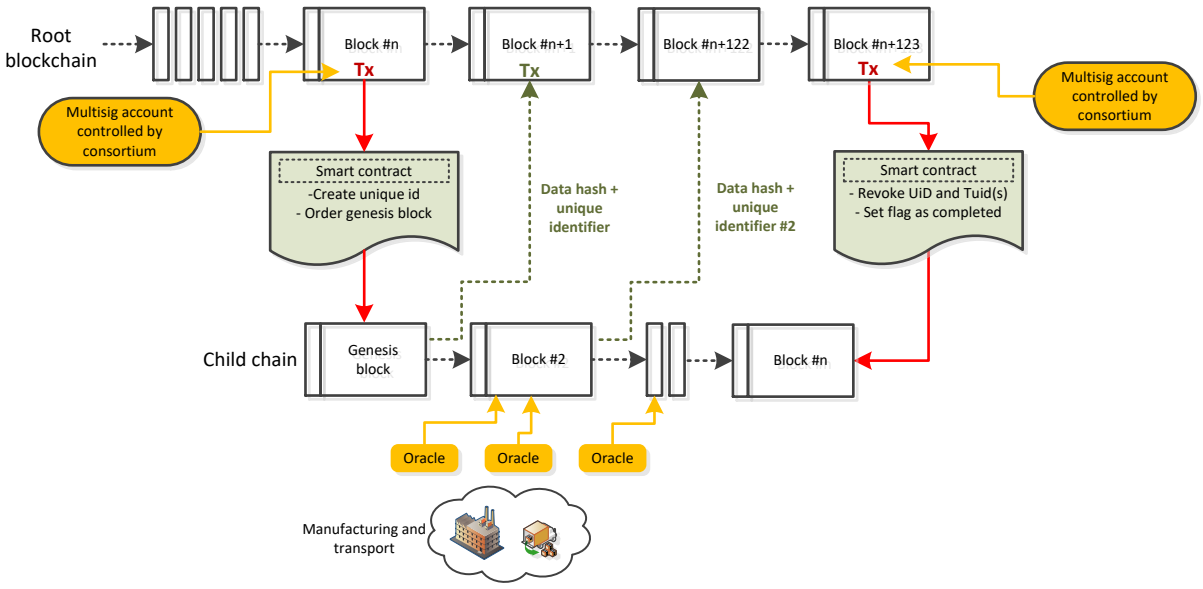

Figure 18: Supply chain use case

We will now describe a case where company A and company B would like to initiate a child chain in order to track the production of smartphone. Company A produces the smartphone and company B handles the transport and sale them directly to consumers. The companies decide to host their own private blockchain using proof of authority as the consensus protocol.

First, they execute the smart contract in the main blockchain from an multi signature account controlled by both of them. They also pass along the digital certificate of their consortium and they set the interval timer to 1 hour. This means that their child chain will update the root state with an hourly interval. The outcome is that their child chain is registered in the main blockchain and they receive the genesis block.

The companies begin using their child chain. There, they register every step of the production using oracles and sensors to track the data. This happens during the manufacturing process of

company A and during the transportation process of company B. Every smartphone produced by this consortium can be traced in this child chain.

Every hour, as set in the beginning, the child chain updates the main blockchain. In this way, the execution happens in the child chain, but also reported in the main blockchain. After the collaboration is finished, the companies decide to revoke the child chain. This happens by executing the proper function in the main blockchain.

For the end user, this means that they can track the smartphone manufacturing and transportation journey from the beginning up to the moment of the purchase. The user can check if the transactions occurred in the blockchain of the specific companies by looking up the unique ids and multi signature account associated with the consortium. They can also check if the child chain is still running or has been revoked.

5. Comparison to existing layer-2 protocols

5.1. Scalability analysis

The scalability of our protocol is similar to sidechains and rollups. The child chains can be setup with different characteristics regarding transaction fees, execution times or consensus protocols. The child chains can have their own Virtual Machine in order to create other child chains or facilitate the execution of dApps. The translates into lower fees compared to the main blockchain and more transactions per second, according to capacity needs. Also, since the child chain can also be a private or permissioned blockchain, it can be configured to allow only predetermined participants.

5.2. Security analysis

Moreover, since the proposed protocol is a pegged solution including lifecycle management, we believe that the security and auditability is also enhanced compared to other layer-2 solutions. A user can not only monitor and recreate the root state of the child chain, but also be sure that the child operates as expected. The end user can therefore audit the data and the origin of the data of the child chain. In the supply chain example, we mentioned before [98], an end user can trust the product received even if the child chain used in the transactions between a producer and the distributor or the distributor and supermarket is a private one. This is because the creation and maintenance of the child chain belonging to these companies is listed in a main blockchain like Ethereum, and the data of these private blockchains can also be verified due to the pegged mechanism. Also, a difference to existing side chain solutions is that the user can also verified that the child chain used was indeed the proper one by validating the unique ids.

The protocol can also support dispute mechanisms in a similar manner to sidechains and rollups. Every node of the child chain can issue a dispute to the main blockchain and this can be resolved by recreating the root state. Since the protocol supports hierarchical multilayers, the upper-level child chain can also function as the dispute mechanism for the lower-level child chains. Another option would be only the main blockchain can resolve the disputes, if the trust level of the child chains is low.

Another aspect is the relation to layer-2 scalability solutions. If the main blockchain uses sharding, then an option could be to distribute the smart contracts which create the child chains in multiple shards. This could potentially create shards with an application specific nature. For example, a number of shards can host gaming child chains, or enterprise applications child chains, or financial child chains. This would allow child chain operators to choose the specific shard for their implementation. Likewise, the users would be able to check if an application is run on a child chain from the correct shard and the correct child chain operator.

5.3. Limitations and considerations of the model

A security issue arises with the multilayer model and the situation where a level-(n) child chain is compromised and the level-(n+1) child chains are also restricted. This is directly related to the hierarchical trust model on which our protocol is based upon. If the main blockchain instructs the revocation of a child chain, then the additional nested child chains will not be able to produce their root state updates as the pegging mechanism dictates. A possible solution to this would be to be able to contact directly the main blockchain and issue a dispute so that the transactions can be reversed and possible locked funds to be returned to the proper owners.

A specific consideration is the creation of forks. The longest chain paradigm indicates that the fork with the longest chain wins the election process according to the consensus protocol. If there is a fork in the child chain then at least two state root updates could be produced in order to be included in the main blockchain. The same applies if there is a fork in the subsequent level-(n+1) child chains. This could be resolved by registering both state roots in the main blockchain until the fork is resolved, after which the correct state update would be kept. However, during this time, a user would not be able to trust which version of the child chain is correct and would have to wait until the fork is resolved.

A final consideration is the use of incentives. If a child chain operates in using proof-of-work consensus protocol, it could potentially not provide the motivation for miners to participate. This is due to the characteristics of the model where the child chains provide scalability by increased throughput but also lower fees. The incentives model is then depended on the child chain operator and the characteristics applied.

5.4. General comments

If we use the layer 2 solutions from the baseline comparison table of the previous chapter and add our model, we see the following:

Table 3: Baseline comparison table including our model

Name	**Layer**	**Main functionality**	**Scaling method**	**Security aspects**	**Example implementations**	**Comments**	**Scalability trilemma**
State channels	2	Use of peer-to-peer channels outside the main chain for transactions between parties.	Within this channel, there can be multiple transactions executed, without using capacity of the main chain.	Enhanced privacy 'Liveliness' requirement of the participants.	Celer Network, Lightning Network, Trinity, Raiden Network	Users of the channel must remain online. They exist temporarily.	Scalability: + Decentralization: Equal Security: + -
Sidechains	2	Side or child chains pegged to main chain.	The sidechain can be designed according to throughput needs.	Security is not ensured by the main chain.	Plasma, Cumulus.	Permanent in nature.	Scalability: + Decentralization: Equal Security: -/+
Roll-ups	2	Computation of transactions is moved off chain, while data remain on chain.	Transactions are executed off chain and only the state root is published in the main chain. Compression of the data used also enhance scalability.	Data is kept in the main chain, making it easy to audit the execution of the transactions.	Loopring, zkTube, Aztec, Starkware, zkSync, Optimism, Arbitrum		Scalability: + Decentralization: - Security: Equal
Our model	2	Transaction execution happens off chain, but validation data remain on chain. Lifecycle management of child chain increases end user trust.	Off chain transaction execution, in a same manner as rollups. Data posted to main chain can be compressed and only a hashed fingerprint could be posted.	Security is ensured by main chain, including the data but also the child chain itself. Child chain ownership can also be proved on the main chain.	-	Can be permanent or temporary in nature.	Scalability: + Decentralization: Equal Security: +

The scores of the scalability trilemma have been set with the following arguments. Scalability is enhanced due to the fact that child chains can take up the load of the main chain. This allows the main chain to provide mainly attestation services for the transactions which are executed in the child chain.

The decentralization is set to equal even if the child chain can also be a privately operated blockchain. Despite that, the transaction and block data are registered in the main blockchain, and the disputes are controlled by the main blockchain as well. This creates a digital environment where even if a corporation runs a child chain to execute transactions, it still needs to attest to the rules of the main public blockchain.

The security is enhanced due to the auditability, lifecycle management and proof of child chain operator ownership features of our model. Since the child chain operator can be proved in the main blockchain and the unique identifier is associated with this child chain, the end user can be assured about the application which runs on this specific child chain. Also, the user can also verify that the child chain is still active and produce new blocks, by checking the relevant root state update on the main blockchain.

As a final remark, we believe that this protocol can prove to be a suitable solution for enterprises and governmental agencies to use and offer blockchain based services. This is due to the enhanced auditability mechanism and use of lifecycle principles, which are features which appeal to this kind of entities. An example could be an agency like the tax service, to use a public blockchain like Ethereum and from there order the creation of other child chains which would offer tax services. In this case the nodes of the child chain could be under the influence of the tax service but could also be independent. The tax service would still be able to audit the data produced by the child chain. The users would still check the validity of the private blockchain using the auditability mechanisms in the main blockchain.

6. Conclusions and future work

In this thesis we focused on the scalability challenge of blockchain implementations, which is related to the scalability trilemma. We identified the prominent layer-1 and layer-2 solutions after making a deep dive in the basic concepts of blockchain. Our survey concluded that the combination of shards and rollups and/or child chains can produce the best balance between scalability and security. We also identified security issues about auditability and trust, which become more severe if private or permissioned blockchains are used. This creates an environment where the users do not trust the operators of these child chains, the data from which are produced, and the services offered from these implementations.

The next part on the thesis focused on the introduction of our protocol. Using this protocol, a child chain can be created, maintained and deleted by a main blockchain. The child chain and main blockchain also are connected with a pegged mechanism ensuring the data auditability and for security reasons. A hierarchical model is also supported where child chains can act as main blockchains for nested child chains. This is using the principles of public key infrastructure where the higher-level entity signs digital certificates for the intermediate level, and this in turn signs digital certificates for the lower-level entities.

Future work includes the functional design of our protocol. This includes the diagrams with the flows between the main blockchain and the child chains, the structure of the smart which controls the operations, the root state update mechanism and the dispute procedures. The details of these steps would provide the documentation for operators which want to participate or consume services based on our protocol.

Moreover, a functional demo could serve as the minimum viable product to better showcase the features of the protocol and identify the errors. This would form the alpha testing phase where entities could be invited to tryout the services in a lab or closed environment.

Bibliography

[1] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System.” [Online]. Available: www.bitcoin.org

[2] N. Moustakas, “EP3695360 - BLOCKCHAIN WITH TRANSACTION CANCELLATION,” 2017. https://register.epo.org/application?number=EP18779691 (accessed Nov. 07, 2021).

[3] G. Ateniese, B. Magri, D. Venturi, and E. R. Andrade, “Redactable Blockchain - Or - Rewriting History in Bitcoin and Friends,” in *Proceedings - 2nd IEEE European Symposium on Security and Privacy, EuroS and P 2017*, Jun. 2017, pp. 111–126. doi: 10.1109/EuroSP.2017.37.

[4] Y. Mesengiser and N. Miloslavskaya, “Problems of Using Redactable Blockchain Technology,” *Procedia computer science*, vol. 190, pp. 582–589, 2021, doi: 10.1016/j.procs.2021.06.068.

[5] C. Dwork and M. Naor, “Pricing via Processing or Combatting Junk Mail,” in *Advances in Cryptology — CRYPTO’ 92*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 139–147. doi: 10.1007/3-540-48071-4_10.

[6] “Proof-of-stake (PoS) | ethereum.org.” https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/ (accessed Oct. 27, 2021).

[7] M. H. Suwito, Y. Ueshige, and K. Sakurai, “Evolution of Bulletin Board & its application to E-Voting-A survey.”

[8] R. Kalis and A. Belloum, “Validating data integrity with blockchain.” [Online]. Available: https://isis.apache.org/

[9] D. Bumblauskas, A. Mann, B. Dugan, and J. Rittmer, “A blockchain use case in food distribution: Do you know where your food has been?,” *International journal of information management*, vol. 52, p. 102008, 2020, doi: 10.1016/j.ijinfomgt.2019.09.004.

[10] “Ethereum Whitepaper | ethereum.org.” https://ethereum.org/en/whitepaper/ (accessed Oct. 26, 2021).

[11] N. Szabom, “Smart Contracts,” 1994. https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html (accessed Oct. 15, 2021).

[12] K. Wu, Y. Ma, G. Huang, and X. Liu, “A First Look at Blockchain-based Decentralized Applications,” Sep. 2019, [Online]. Available: http://arxiv.org/abs/1909.00939

[13] F. A. Aponte-Novoa, A. L. S. Orozco, R. Villanueva-Polanco, and P. Wightman, “The 51% Attack on Blockchains: A Mining Behavior Study,” *IEEE Access*, vol. 9, pp. 140549–140564, 2021, doi: 10.1109/ACCESS.2021.3119291.

[14] H. Wang, Q. Yan, and V. C. M. Leung, “The impact of propagation delay to different selfish miners in proof-of-work blockchains,” *Peer-to-Peer Networking and Applications*, vol. 14, no. 5, pp. 2735–2742, Sep. 2021, doi: 10.1007/s12083-021-01087-5.

[15] M. Neuder, D. J. Moroz, R. Rao, and D. C. Parkes, “Low-cost attacks on Ethereum 2.0 by sub-1/3 stakeholders,” Feb. 2021, [Online]. Available: http://arxiv.org/abs/2102.02247

[16] J. J. Kearney and C. A. Perez-Delgado, “Vulnerability of Blockchain Technologies to Quantum Attacks,” May 2021, doi: 10.1016/j.array.2021.100065.

[17] I. Bashir, *Mastering Blockchain : Distributed ledger technology, decentralization, and smart contracts explained, 2nd Edition*.

[18] “Blocks | ethereum.org.” https://ethereum.org/en/developers/docs/blocks/ (accessed Oct. 26, 2021).

[19] “Why sharding is great: demystifying the technical properties.” https://vitalik.ca/general/2021/04/07/sharding.html (accessed Nov. 09, 2021).

[20] Erik Gregersen, “BitTorrent,” *Encyclopædia Britannica Online*. Encyclopædia Britannica Inc, 2020.

[21] H. Halpin, “Deconstructing the Decentralization Trilemma,” 2020. doi: 10.5220/0009892405050512.

[22] “Binance Smart Chain: A Parallel Binance Chain to Enable Smart Contracts.” https://www.binance.org/en/smartChain (accessed Nov. 09, 2021).

[23] “An Introduction to Binance Smart Chain (BSC) | Binance Academy.” https://academy.binance.com/en/articles/an-introduction-to-binance-smart-chain-bsc (accessed Nov. 09, 2021).

[24] “Messari researchers slam Binance Smart Chain over centralized validators.” https://cointelegraph.com/news/messari-researchers-slam-binance-smart-chain-over-centralized-validators (accessed Nov. 09, 2021).

[25] D. van de Ruit and N. Moustakas, “EP3499789 - PRIMARY AND SECONDARY BLOCKCHAIN DEVICE ,” 2017 Accessed: Oct. 13, 2021. [Online]. Available: https://register.epo.org/application?number=EP18212902

[26] D. van de Ruit and N. Moustakas, “ EP3698518 - PRIMARY AND SECONDARY BLOCKCHAIN DEVICE,” 2017 Accessed: Oct. 13, 2021. [Online]. Available: https://register.epo.org/application?number=EP18786794

[27] J. van den Berg *et al.*, “On (the Emergence of) Cyber Security Science and its Challenges for Cyber Security Education,” 2015.

[28] A. E. Gencer, S. Basu, I. Eyal, R. van Renesse, and E. G. Sirer, “Decentralization in Bitcoin and Ethereum Networks”.

[29] K. Raj, *Foundations of Blockchain*, 1st edition. 2019.

[30] A. Antonopoulos and G. Wood, “Mastering Ethereum BUILDING SMART CONTRACTS AND DAPPS,” 2019. [Online]. Available: www.EBooksWorld.ir

[31] C. Research, “STANDARDS FOR EFFICIENT CRYPTOGRAPHY SEC 2: Recommended Elliptic Curve Domain Parameters,” 2000.

[32] C. Paar and J. Pelzl, “Understanding Cryptography: A Textbook for Students and Practitioners.”

[33] A. Antonopoulos, “Mastering Bitcoin Programming the open blockchain,” 2017.

[34] D. Johnson, A. Menezes, and S. Vanstone, “The Elliptic Curve Digital Signature Algorithm (ECDSA),” *International journal of information security*, vol. 1, no. 1, pp. 36–63, 2001, doi: 10.1007/s102070100002.

[35] S. Levy, “Performance and Security of ECDSA,” 2015, Accessed: Nov. 14, 2021. [Online]. Available: http://arxiv.

[36] K. Kulkarni, *Learn Bitcoin and Blockchain*. Birmingham: Packt Publishing, Limited, 2018.

[37] “Oracles | ethereum.org.” https://ethereum.org/en/developers/docs/oracles/ (accessed Nov. 14, 2021).

[38] I. A. Omar, H. R. Hasan, R. Jayaraman, K. Salah, and M. Omar, “Implementing decentralized auctions using blockchain smart contracts,” *Technological forecasting & social change*, vol. 168, p. 120786, 2021, doi: 10.1016/j.techfore.2021.120786.

[39] B. Lashkari and P. Musilek, “A Comprehensive Review of Blockchain Consensus Mechanisms,” *IEEE access*, vol. 9, pp. 43620–43652, 2021, doi: 10.1109/ACCESS.2021.3065880.

[40] “Global Payment Solutions - Instant Processing | Ripple.” https://ripple.com/ (accessed Nov. 15, 2021).

[41] “Hyperledger Fabric – Hyperledger Foundation.” https://www.hyperledger.org/use/fabric (accessed Nov. 15, 2021).

[42] M. Dabbagh, K.-K. R. Choo, A. Beheshti, M. Tahir, and N. S. Safa, “A survey of empirical performance evaluation of permissioned blockchain platforms: Challenges and opportunities,” *Computers & security*, vol. 100, 2021, doi: 10.1016/j.cose.2020.102078.

[43] “Blockchain Technology and Its Potential Impact on the Audit and Assurance Profession”.

[44] “Ethereum Virtual Machine (EVM) | ethereum.org.” https://ethereum.org/en/developers/docs/evm/ (accessed Nov. 10, 2021).

[45] “patricia-tree | Ethereum Wiki.” https://eth.wiki/en/fundamentals/patricia-tree (accessed Nov. 10, 2021).

[46] “Introduction to smart contracts | ethereum.org.” https://ethereum.org/en/developers/docs/smart-contracts/ (accessed Nov. 10, 2021).

[47] “Ethereum Virtual Machine Opcodes.” https://www.ethervm.io/ (accessed Nov. 10, 2021).

[48] “Remix - Ethereum IDE.” https://remix.ethereum.org/ (accessed Nov. 10, 2021).

[49] “Solidity Programming Language | The Solidity language portal is a comprehensive information page for the Solidity programming language. It features documentation, binaries, blog, resources & more.” https://soliditylang.org/ (accessed Nov. 10, 2021).

[50] “Smart contract languages | ethereum.org.” https://ethereum.org/en/developers/docs/smart-contracts/languages/ (accessed Nov. 14, 2021).

[51] R. Modi, *Solidity Programming Essentials*. Birmingham: Packt Publishing, Limited, 2018.

[52] “Introduction to dapps | ethereum.org.” https://ethereum.org/en/developers/docs/dapps/ (accessed Nov. 10, 2021).

[53] “GitHub:bitcoin/consensus.h line 10.” https://github.com/bitcoin/bitcoin/blob/3038eb63e8a674b4818cb5d5e461f1ccf4b2932f/src/consensus/consensus.h#L10 (accessed Nov. 07, 2021).

[54] “Bitcoin Average Transactions Per Block.” https://ycharts.com/indicators/bitcoin_average_transactions_per_block (accessed Nov. 08, 2021).

[55] “GitHub - bitcoin/bips: Bitcoin Improvement Proposals.” https://github.com/bitcoin/bips (accessed Nov. 10, 2021).

[56] “bips/bip-0101.mediawiki at master · bitcoin/bips · GitHub.” https://github.com/bitcoin/bips/blob/master/bip-0101.mediawiki (accessed Nov. 10, 2021).

[57] “bips/bip-0103.mediawiki at master · bitcoin/bips · GitHub.” https://github.com/bitcoin/bips/blob/master/bip-0103.mediawiki (accessed Nov. 10, 2021).

[58] “bips/bip-0109.mediawiki at master · bitcoin/bips · GitHub.” https://github.com/bitcoin/bips/blob/master/bip-0109.mediawiki (accessed Nov. 10, 2021).

[59] “Four Key Disagreements Between Bitcoin Classic and Bitcoin Core - Bitcoin Magazine: Bitcoin News, Articles, Charts, and Guides.” https://bitcoinmagazine.com/technical/four-key-disagreements-between-bitcoin-classic-and-bitcoin-core-four-key-disagreements-between-bitcoin-classic-and-bitcoin-core-four-key-disagreements-between-bitcoin-classic-and-bitcoin-core-14571067 (accessed Nov. 10, 2021).

[60] I. Eyal, A. Efe, G. Emin, G. Sirer, and R. van Renesse, “Bitcoin-NG: A Scalable Blockchain Protocol”.

[61] M. Vukolić, “The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication,” in *Open Problems in Network Security*, 2016, pp. 112–125. doi: 10.1007/978-3-319-39028-4_9.

[62] W. da R. Franca, “MongoDB data modeling : focus on data usage and better design schemas with the help of MongoDB, chapter 7,” 1st edition., 2015.

[63] G. Yu, X. Wang, K. Yu, W. Ni, J. A. Zhang, and R. P. Liu, “Survey: Sharding in Blockchains,” *IEEE Access*, vol. 8, pp. 14155–14181, 2020, doi: 10.1109/ACCESS.2020.2965147.

[64] “Shard chains | ethereum.org.” https://ethereum.org/en/eth2/shard-chains/ (accessed Nov. 15, 2021).

[65] “Sharding-FAQs | Ethereum Wiki.” https://eth.wiki/sharding/Sharding-FAQs (accessed Nov. 15, 2021).

[66] M. Zamani, M. Movahedi, and M. Raykova, “Rapid-Chain: Scaling Blockchain via Full Sharding,” p. 18, doi: 10.1145/3243734.3243853.

[67] M. Muneeb, H. Pervez, M. Usama Irfan, and I. Ul Haq, “A Comparative Analysis of DAG-Based Blockchain Architectures,” 2018, doi: 10.1109/ICOSST.2018.8632193.

[68] “Home | IOTA.” https://www.iota.org/ (accessed Nov. 13, 2021).

[69] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, “SPECTRE: A Fast and Scalable Cryptocurrency Protocol”.

[70] “Nano | Digital money for the modern world.” https://nano.org/ (accessed Nov. 13, 2021).

[71] “Ethereum : Security vulnerabilities.” https://www.cvedetails.com/vulnerability-list/vendor_id-17524/Ethereum.html (accessed Nov. 15, 2021).

[72] “Bitcoin : Security vulnerabilities.” https://www.cvedetails.com/vulnerability-list/vendor_id-12094/Bitcoin.html (accessed Nov. 15, 2021).

[73] M. Mirkin, Y. Ji, J. Pang, A. Klages-Mundt, I. Eyal, and A. Juels, “BDoS: Blockchain Denial-of-Service,” 2019.

[74] “Seeming oracle attack causes $100m in Ethereum DeFi liquidations | CryptoSlate.” https://cryptoslate.com/seeming-oracle-attack-causes-100m-in-ethereum-defi-liquidations (accessed Nov. 15, 2021).

[75] S. Sayeed, H. Marco-Gisbert, and T. Caira, “Smart Contract: Attacks and Protections,” *IEEE access*, vol. 8, pp. 24416–24427, 2020, doi: 10.1109/ACCESS.2020.2970495.

[76] N. F. Samreen and M. H. Alalfi, “Reentrancy Vulnerability Identification in Ethereum Smart Contracts,” 2021. doi: 10.1109/IWBOSE50093.2020.9050260.

[77] Y. Huang, Y. Bian, R. Li, J. L. Zhao, and P. Shi, “Smart contract security: A software lifecycle perspective,” *IEEE Access*, vol. 7. Institute of Electrical and Electronics Engineers Inc., pp. 150184–150202, 2019. doi: 10.1109/ACCESS.2019.2946988.

[78] Z. Wang, H. Jin, W. Dai, K.-K. R. Choo, and D. Zou, “Ethereum smart contract security research: survey and future research opportunities,” *Frontiers of Computer Science*, vol. 15, no. 2, 2020, doi: 10.1007/s11704-020-9284-9.

[79] P. Praitheeshan, L. Pan, J. Yu, J. Liu, and R. Doss, “Security Analysis Methods on Ethereum Smart Contract Vulnerabilities: A Survey,” Aug. 2019, [Online]. Available: http://arxiv.org/abs/1908.08605

[80] X. Sun, S. Lin, V. Sjöberg, and J. Jie, “How to Exploit a DeFi Project,” in *Financial Cryptography and Data Security. FC 2021 International Workshops*, Berlin,

Heidelberg: Springer Berlin Heidelberg, 2021, pp. 162–167. doi: 10.1007/978-3-662-63958-0_14.

[81] "The Story of the DAO — Its History and Consequences | by Samuel Falkon | The Startup | Medium." https://medium.com/swlh/the-story-of-the-dao-its-history-and-consequences-71e6a8a551ee (accessed Nov. 06, 2021).

[82] "Ethereum Blocks #1920000 | Etherscan." https://etherscan.io/block/1920000 (accessed Dec. 05, 2021).

[83] "History and Forks of Ethereum | ethereum.org." https://ethereum.org/en/history/#dao-fork (accessed Dec. 05, 2021).

[84] "Ethereum Classic." https://ethereumclassic.org/ (accessed Dec. 05, 2021).

[85] "Program the Blockchain | State Channels for Two-Player Games." https://programtheblockchain.com/posts/2018/05/11/state-channels-for-two-player-games/ (accessed Dec. 05, 2021).

[86] "Lightning Network." https://lightning.network/ (accessed Dec. 05, 2021).

[87] A. Back *et al.*, "Enabling Blockchain Innovations with Pegged Sidechains," 2014.

[88] A. K. Koç, E. Yavuz, U. C. Çabuk, and G. Dalkiliç, "Towards secure e-voting using ethereum blockchain," *6th International Symposium on Digital Forensic and Security, ISDFS 2018 - Proceeding*, vol. 2018-January, pp. 1–6, May 2018, doi: 10.1109/ISDFS.2018.8355340.

[89] F. Gai, J. Niu, C. Grajales, M. M. Jalalzai, and C. Feng, "Cumulus: A BFT-based Sidechain Protocol for Off-chain Scaling".

[90] A. Singh, K. Click, R. M. Parizi, Q. Zhang, A. Dehghantanha, and K. K. R. Choo, "Sidechain technologies in blockchain networks: An examination and state-of-the-art review," *Journal of Network and Computer Applications*, vol. 149, p. 102471, Jan. 2020, doi: 10.1016/J.JNCA.2019.102471.

[91] J. Poon and V. Buterin, "Plasma: Scalable Autonomous Smart Contracts," 2017, Accessed: Dec. 08, 2021. [Online]. Available: https://plasma.io/

[92] “A rollup-centric ethereum roadmap - Fellowship of Ethereum Magicians.” https://ethereum-magicians.org/t/a-rollup-centric-ethereum-roadmap/4698 (accessed Dec. 08, 2021).

[93] “An Incomplete Guide to Rollups.” https://vitalik.ca/general/2021/01/05/rollup.html (accessed Dec. 05, 2021).

[94] “Optimism - Ethereum at Lower Costs & Lightning Speed.” https://www.optimism.io/ (accessed Dec. 05, 2021).

[95] M. Kędziora, D. Pieprzka, I. Jóźwiak, Y. Liu, and H. Song, “Analysis of Segregated Witness Implementation for Increasing Efficiency and Security of the Bitcoin Cryptocurrency,” in *Computational Collective Intelligence*, Cham: Springer International Publishing, 2020, pp. 640–651. doi: 10.1007/978-3-030-63007-2_50.

[96] S. Li, M. Yu, C.-S. Yang, A. S. Avestimehr, S. Kannan, and P. Viswanath, “PolyShard: Coded Sharding Achieves Linearly Scaling Efficiency and Security Simultaneously”.

[97] Forrester, “Seize The Day: Public Blockchain Is On The Horizon Get started Examine The Limitations Of Private Blockchain And Benefits Of Public Blockchain To Make The Most Of This Technology Opportunity FORRESTER OPPORTUNITY SNAPSHOT: A CUSTOM STUDY COMMISSIONED BY EY |,” 2019.

[98] F. Casino *et al.*, “Blockchain-based food supply chain traceability: a case study in the dairy sector,” *International journal of production research*, vol. 59, no. 19, pp. 5758–5770, 2021, doi: 10.1080/00207543.2020.1789238.

[99] T. Spies, “Chapter 3 - Public Key Infrastructure,” in *Cyber Security and IT Infrastructure Protection*, Elsevier Inc, 2014, pp. 75–107. doi: 10.1016/B978-0-12-416681-3.00003-3.