Improving blockchain scalability using layer 2 protocols

REPORT

Sanskar Srivastava  
CSE  
Vellore Institute of TechnologyChennai, India

Rishabh Varshney  
CSE  
Vellore Institute of TechnologyChennai, India

Sagar Pahuja

CSE  
Vellore Institute of TechnologyChennai, India

*Abstract*— In the modern world most of our sensitive data is stored online and transactions take place through the web. The growing threat of hackers and attacks along with the need for a more trustworthy system led to the introduction of blockchain and cryptocurrency. Blockchain in the technical sense is a digitized, de-centralized, public ledger of all the transactions. It is a shared immutable ledger that allows the recording and tracking of assets. Using blockchain anything of value can be tracked and traded on a secure network massively reducing the risks and costs involved. It has become an important and ideal technology. Since business runs on information transactions need to be as fast as possible while also being trustworthy and maintaining integrity. Blockchain is very useful in this case as it is capable of sharing completely transparent information stored in a ledger that can be accessed only by the acknowledged network members. Even with such advantages blockchain is not an infallible technology and has some disadvantages. One such improvement that is necessary is the scalability. Scalability is essentially the capacity of the network to sustain larger transactions. To increase the performance of blockchain various solutions have been proposed and one such popular solution is the implementation of layer 2 protocols.

# Introduction

As we have already learnt blockchain has become an essential technology in various fields like business, finance, government, logistic systems etc. A blockchain network allows tracking of orders, payments, accounts and anything of value while allowing members to share a single point of view which displays the details of the transaction while offering integrity and increasing the speeds of transactions. In a blockchain all the participants have access to the distributed ledger and the record of transactions. This record is immutable and the transactions are only recorded once and cannot be changed again. If there is an error in the transaction then a new transaction has to be created and all of the transactions including the error one is visible. To improve the transaction speed there is a set of predefined rules known as a smart contract which is executed automatically. This contract defines the conditions for the bond transfers and much more. This blockchain is the main system of the current cryptocurrency and transactions. There are many solutions that have been introduced to increase the speed of the transactions of the existing blockchains. One such popular solution is layer 2 protocols. Layer 2 protocols are a secondary framework that is structured on top of the existing blockchain. Essentially the aim is to solve the transaction speed and scalability issues that are being faced by all the major crypto networks. For reference bitcoin and Ethereum are not able capable of processing more than a thousand transactions per second. This is certainly not slow but in terms of long-term improvement it is imperative to increase the throughput. Major examples of this are Bitcoin lightning network/Ethereum Raiden, Ethereum plasma, optimistic rollup. Since the layer 2 protocols create a secondary network through which transactions can take place independently this is also known as an off-chain solution. The advantage in this case is that this does not affect or change the main chain so it can increase the speeds without affecting the network security. To understand the solutions and compare the speeds we will first understand how blockchains work and how each layer 2 protocol functions after which we will examine the advantages and disadvantages of each of them

RELATED WORKS

Searching for trust: blockchain technology in an age of disinformation by Victoria Lemieux says blockchain immutability is best viewed as a sustained commitment that a group of individuals holds onto because they believe that the attribute is desirable but has led to increases in the forgery and manipulation of the evidential basis of factual claims.

Blockchain-empowered data-driven networks: a survey and outlook by Victor Leung says pointing towards a data-driven paradigm to better cater to the explosive growth of mobile services but many important challenges and open issues remain to be addressed before blockchain can be deployed widely to enable future ddns.

Blockchain-enabled edge intelligence for iot: background, emerging trends and open issues by Yao du says this decentralization technology has become a powerful model to establish trust among trust less entities but many iot devices are designed to be battery operated and have a compact physical size; thus, they have very limited energy and computation resources.

How blockchain capabilities advance data sharing and benefit virtual care by Chang lu says the systemic adoption of virtual care means moving beyond third-party video tools that currently come to mind around traditional telehealth services but patients expand the types of encounters and providers they engage in, as well as where their data resides.

A simple explanation of bitcoin and block chain technology by Rl twesegi says any one from all over the world can utilize the block chain technology but since bitcoin transactions are anonymous and unregulated, another disadvantage is the lack of security.

Opportunities and limitations of public blockchain-based supply chain traceability by Oliver Bischoff says blockchain-based supply chain traceability by identifying the opportunities and limitations that accompany the adoption of public blockchains. But blockchain technology is provoking significant disruptions, thereby affecting supply chain management.

Blockchain research, practice and policy: applications, benefits, limitations, emerging research themes and research agenda by Laurie hugus says significant potential to benefit a number of industry wide use cases but has been slow to engender any significant momentum within the information systems (is) and information management (im) literature.

Potential and limits of blockchain technology for networked businesses by Roger Wh says dynamic shared ledgers that can be applied to ensure transparency and traceability and to save time but address some of the deeper technical foundations such as scalability, accountability, and security.

Blockchain technology: limitations and future possibilities by Suvarna Sharma says it is a digital ledger that holds information on transactions taking place over the web but immutability of data.

The advantages and disadvantages of the blockchain technology by Julia Strebko says store information in a decentralized manner so everyone can verify the correctness of the information by using zero-knowledge proof but a blockchain network is vulnerable if one entity owns at least half the nodes.

Blockchain scaling with layer 2: theory and practice by Stepan Gurushin says examines some of the existing solutions of second layer approach to blockchain scaling but scalability, speed and throughput is an inherent problem for any public blockchain

An overview of ethereum layer 2 optimum scaling solutions by Edward de la says they allow for scalability and increased throughput while still holding the integrity of the ethereum blockchain but it can remove liquidity from the primary blockchain.

A survey of layer-two blockchain protocols by Ankit Gangwal says functionalities associated with reduced transaction fees, no restrictions on capacity, and improved transaction speed but there are more potential security and privacy vulnerabilities.

Performance-based analysis of blockchain scalability metric by Ranjana Shevkar says ability of that platform to support increasing load of transactions, as well as increasing the number of nodes in the network but the scalability issue with blockchain emerges mostly when the number of nodes and transactions increases.

Lightning network: a comparative review of transaction fees and data analysis by Rida Khan focusses on lightning network, the off-chain, scalable and high throughput payment solution from bitcoin but scalability and throughput issues plague the technology and prevent its mass scale adoption.

Challenges and emerging solutions for public blockchains by Radu Vasiu says public blockchains are open-source. Transactions are recorded anonymously but high risk of potential collisions.

Systematic literature review of challenges in blockchain scalability by Teejay low says recommended solutions for the blockchain scalability challenge refers to improving consensus protocols the limited capability of the bitcoin network to handle large amounts of transaction data on its platform in a short span of time.

Layer 2 blockchain scaling: a survey by Robert Spadafore says focusing on theoretical and practical aspects and highlighting the main differences among the examined frameworks but it diminishes Ethereum’s composability.

Continuous credit networks and layer 2 blockchains: monotonicity and sampling by Ashish Goel says it enables meaningful liquidity analysis for real-world parameter regimes but continuous case would be harder to examine.

An introduction to plasma by Alexander Butler says executing transactions off-chain with its own mechanism for block validation but long waiting periods fund withdrawal and hence poor experience for certain users.

Lightning network by Vitarik Beutrin says fast, easy and scalable bitcoin transfer.

Introduction to bitcoin payment transfers by Satoshi Nakamoto says better peer to peer transfer speed solution.

Ethereum sharding and its application talks about ethereum sharding with its application.

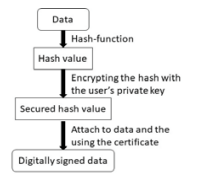
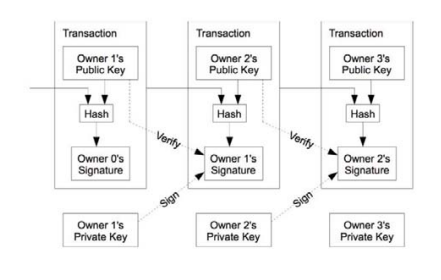
Raiden networks

Atomic transfers

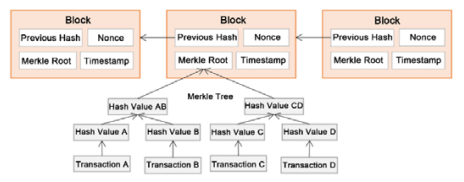
# Blockchain

The basic working of a blockchain can be explained using blocks of data. Whenever a transaction takes place, it is recorded and stored in a block of data. Each block contains the cryptographic hash of the previous block and this information is generated automatically. The data block can record any transaction related information. Each block is connected to the one before and after it forming a linear chain of data. The block also has a timestamp and can confirm the exact time and sequence of the transactions. The blocks being linked in this manner prevent any other block from being altered or inserted. Each new block strengthens the security and verification of the entire blockchain.

The transparency of the blockchain is achieved by the reocrding the transactions and allowing the information to be viewed by all the users at any gien time. The transactions contain information for the EOAs or the externally owned accounts. When the new EOA is created the public-private key is created along with it and stored in the file JSON. The private key of the sender is imprtant and is needed for the sign of the transactions and the public key and the password are important for the transactions to the other accounts.

When a new block is created the time of the block generation is noted and the next block with all transactions is created 120 seconds after the time when the last block was signed by the miner. Block creation is done as shown below



Ethereum is a popular and flexible blockchain platform which is also open to use. The platform is decentralized and establishes a peer-to-peer network that securely executes and verifies the smart contracts. Smart contract is the application code that resides at a specific address on the blockchain known as the contract address. Transactions are sent and received by user-created Ethereum accounts. The sender must sign the transactions and spend ether as a cost of the transactions.

# Layer 2 protocols

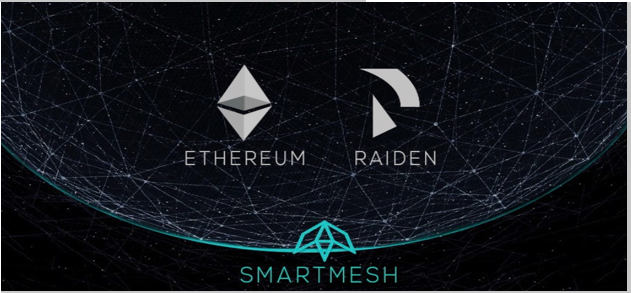
In an ideal scenario the blockchain should be able to handle thousands of transactions per second but currently this is not possible. The difference is due to the level of decentralization and privacy that the blockchains aim to provide. It takes time to accept transactions, then mine distribute and validate them. To allow the blockchains to handle a higher number of transactions per second we use layer 2 scaling solutions. These solutions should posses the same security as the main chain and act as off-chains since they form secondary frameworks for the main chain. The aim is to provide the necessary improvements without needing to increase the block size or include other measures that would hinder the capacity of decentralization and security. Examples of such solutions are –

Lightning network / Raiden: Raiden and its Workings Principles

On top of the Ethereum Blockchain is an infrastructure layer called the Raiden Network. Although the fundamental concept is straightforward, the underlying protocol is rather complicated, making implementation difficult. To construct scalable decentralized apps based on the Raiden Network, developers may communicate using a very easy API after the complexities have been abstracted away.

For transferring ERC20-compliant tokens on the Ethereum blockchain, the Raiden Network offers an off-chain scaling option. It is Ethereum's answer to Bitcoin's Lightning Network, allowing for very instantaneous, inexpensive, scalable, and private payments.

With the exception of an initial one-time on-chain construction and a final channel closure, these transactions may be executed instantly and without involving the blockchain itself. A legally-binding agreement upheld by the Ethereum blockchain is a Raiden balance proof. As long as at least one of the parties agrees to present it to the blockchain, digital signatures ensure that neither party can back out of any of the value transactions included within. A Raiden balance proof is as legally binding as an on-chain transaction since only the two parties have access to the tokens put in the payment channel's smart contract.



The network protocol of Raiden is where its greatest power rests. It becomes impossible to create payment channels between all potential peers since doing so still necessitates on-chain transactions. However, it turns out that if there is at least one path across a network of channels that connects the two parties, you do not require a direct payment channel between a payer and a payee.

Furthermore, unlike on-chain transactions, payment channel transfers are fee-free. Although intermediaries in the larger network will wish to charge modest percentage fees for supplying their own channels to the network, this will result in complicated routing and a crowded market for channel fees.

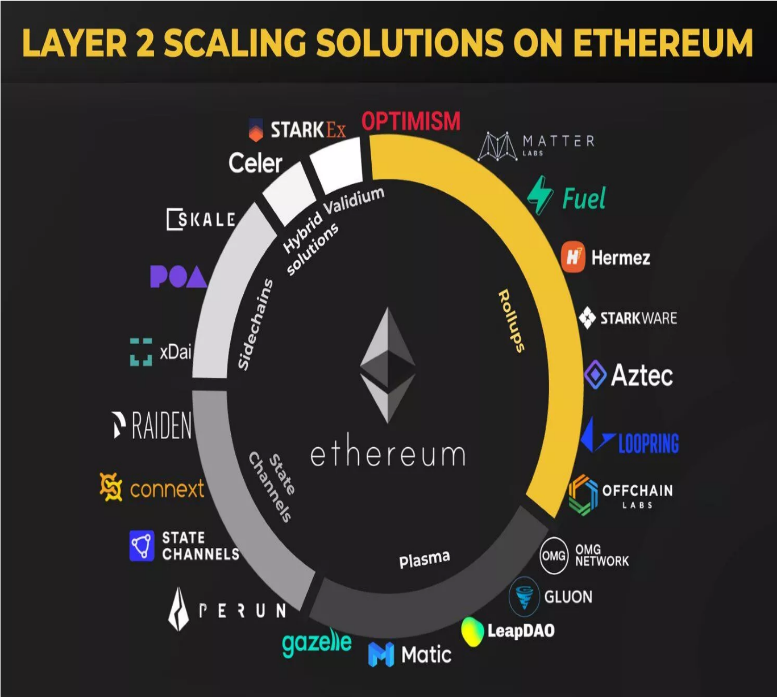
# Benefits of Raiden and uses

Depending on the computational resources needed, every transaction on the Ethereum blockchain has a price. As a result, whether transferring ERC20 tokens or Ether itself, fees are generally independent of the actual amount of value transmitted. On-chain transactions are thus better suited for medium- to large-scale value transfers but less so for smaller-scale transactions involving a few dollars or even pennies.

It makes no difference whether a transfer is sent all at once or broken up into thousands of tiny instalments. On the Raiden Network, almost no transfer is too small to be sent effectively.

Raiden transfers are also immediate in the sense that you may be confident the transferred value is now yours as soon as you get an off-chain Raiden transfer. On-chain transfers, however, must be confirmed within a block of time, which is determined by how long it takes miners to select your transaction from the pool of outstanding transactions. With Raiden transfers, you can send, receive, and confirm transfers as quickly as sending an online chat message rather than having to wait for the following blocks to do so.

In addition to costs, blockchains also have a scalability issue that Raiden aids in addressing. Regardless of the number of users, the majority of contemporary blockchains have a set or semi-fixed capacity limit. In striking contrast, the Raiden Network's capacity expands linearly with the number of users, creating a decentralised transfer network that is effective and secure for the future.

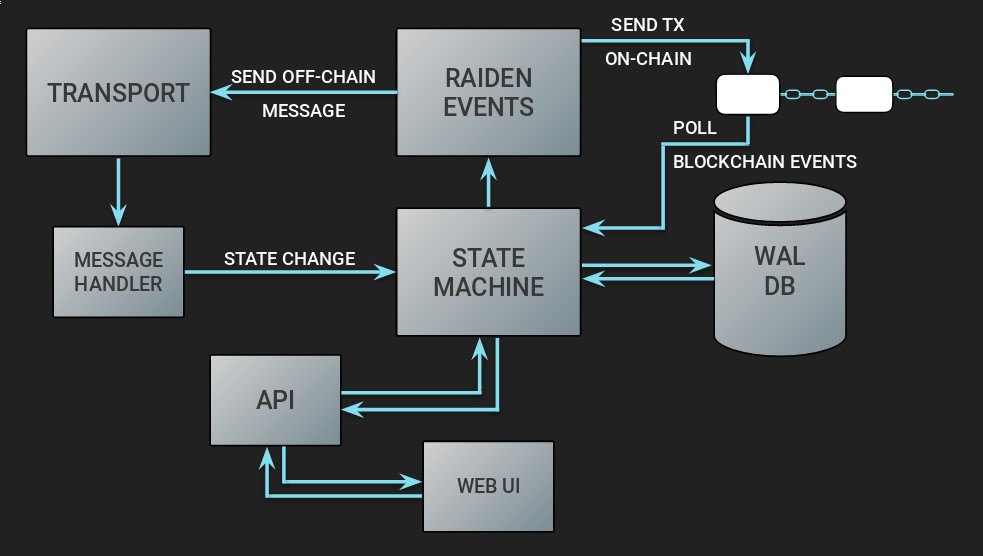


# Architecture

Tokens must be secured in a smart contract for the duration of the payment channel in order to guarantee that participants pay their debts. This deposit guarantees that until the channel is eventually closed by either participant, tokens may only be used to send and receive tokens to and from the channel partner, preventing both parties from double spending their tokens to other peers.

Participants may freely exchange what are essentially certified checks after a channel has been established. However, each peer simply maintains a copy of the most recent check and does not maintain a record of all checks. The ultimate total of all Raiden transfers delivered to a participant up to a specific moment is included in the balance proof and is digitally signed by the sender.

As we have already explained the working of Raiden network with an example in review 1.



# Mediation Fees

By leveraging a network of interconnected payment channels as a payment intermediary, Raiden enables users to send money to anybody in the network. The nodes on the chosen payment channel between the initiator and target, known as mediating nodes, are eligible to receive mediation fees.

The initiator pays the mediation costs by slightly raising the token total over that intended for the target. Additionally, this implies that mediation fees are always paid in tokens of the same sort as the transfer.

# Benefits Of Mediation Fees

The following factors make the payment network more robust:

users being encouraged to operate mediating nodes and choose to use routes with cheaper fees in order to balance the related payment channels

A strong payment network enables consumers to make affordable and dependable payments. Users must pay mediation costs when making payments, although these fees ultimately serve their own benefits.

# Mediation Fees Calculation

Each mediator may select their own price structure for payments. This charge schedule is divided into three sections: a fixed cost per mediation a charge based on the number of tokens mediated

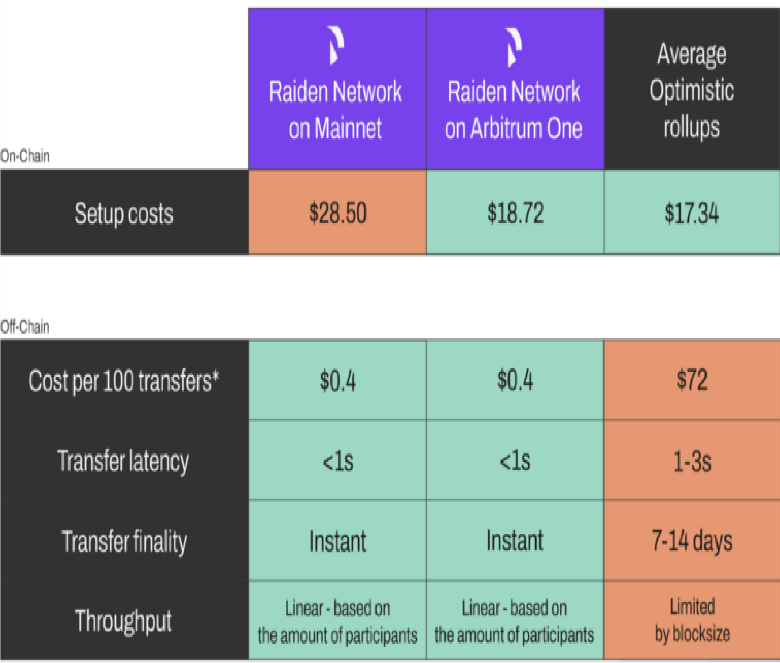
a channel-capacity-exhausting imbalance cost that rises as the payment is made, thereby prohibiting the usage of the channel for more payments

The total of these costs denotes the charge for a single mediator. The total cost of mediation for a payment is determined by adding the costs for each mediator along the payment route.

## Graphs and Increased Transaction Speed

Raiden is focused on instant transaction speeds and lower costs per transaction.

Lower costs per transaction and instant transfer speeds compared to the mainchain of Ethereum.



## Limitations of Raiden Network and Future Plans

Some of your tokens must be locked up in a smart contract for the duration of the payment channel in order to use NE Raiden transactions. You would not want to lock up too much value in a payment channel, just as you wouldn't want to take more than minimal sums of cash from an ATM. After making an ATM withdrawal, you are unable to utilise the money again for wire transfers or online purchases. Similarly, payment channel deposits are anticipated to be quite tiny, making it challenging to move significant sums of tokens through the network of channels. This is due to the likelihood that each participant in the network will have numerous channels open at any given moment.

To save on the additional expense of channel lifetime maintenance and to eliminate the requirement for routing via typically underwhelming payment channels, large value transactions should still be carried out directly on the blockchain.

## Privacy and Security Analysis

Transfers are essentially private with the Raiden protocol since the majority of the work is done off-chain. Until participants settle and remove their cash and the net channel balance is made public, channel balances are not visible to the general public. When they do, it may be exceedingly challenging to link on-chain transactions to off-chain Raiden transfers since channel balances may have previously been obscured by earlier intermediate transfers that went through this and other channels connected to the same nodes. In order to maximize the amount of anonymity, users may even provide paid services to artificially rebalance and obscure channel balances. Raiden will take care to safeguard traffic and sensitive data sent through the network on the messaging layer.

The messaging service will shield users' IP addresses from prying eyes, guarding against DoS assaults on random nodes. Furthermore, pre-computed routes could employ an onion routing protocol, which prevents intermediary nodes from knowing the target address of a Raiden transfer. Only the next channel in the route is disclosed to each participant by the protocol.

## Conclusion on Raiden Network

The Raiden Network establishes direct connections between members using bidirectional token payment mechanisms. Additionally, rather than attempting to link each participant directly, it offers a mechanism to relay token transfers across routes of channels in order to take use of the natural channel network structure. To guarantee that a mediated transfer is accepted by all parties, multihop transfers are safeguarded using cryptographic hash locks.

The Raiden Network seeks to expand existing on-chain restrictions by using the technologies to offer almost quick, inexpensive, scalable, and private payments based on Ethereum ERC20 tokens.

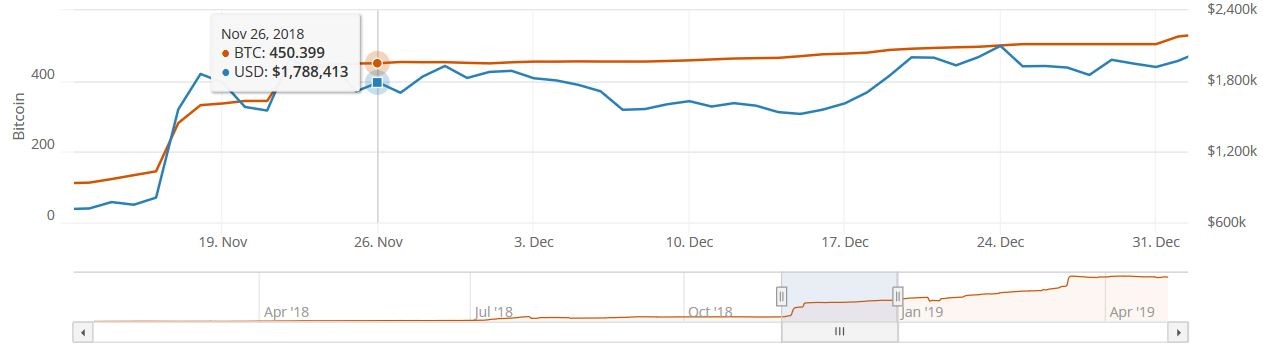
## Performance metric Graphs

1. Increase in number of transactions using lightning network

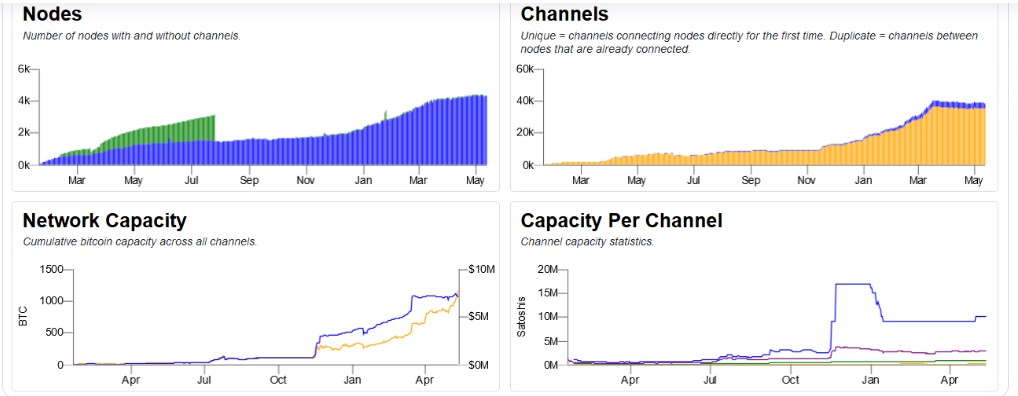
## 2. Valuation Increase with usage of layer 2 protocol



## 3. Increase in capacity growth By over 300% within two weeks of implementation(in Bitcoin)



1. Increase in node capacity and network capacity for transfers.



1. Ethereum Transaction increases after layer 2 protocol implementation.



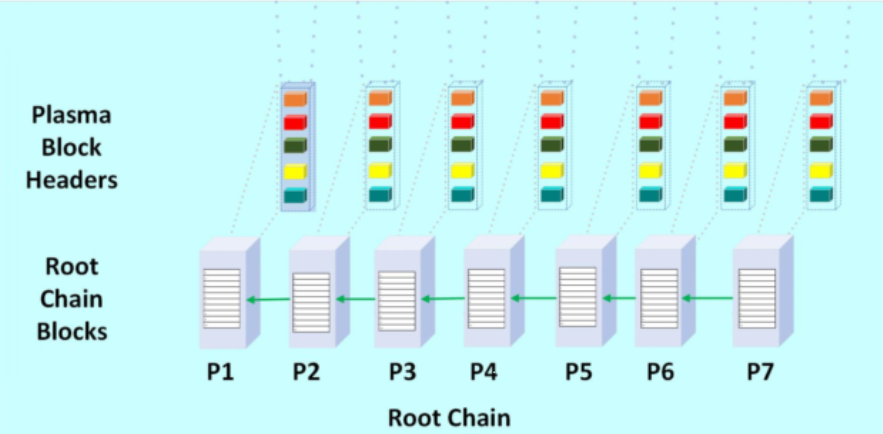
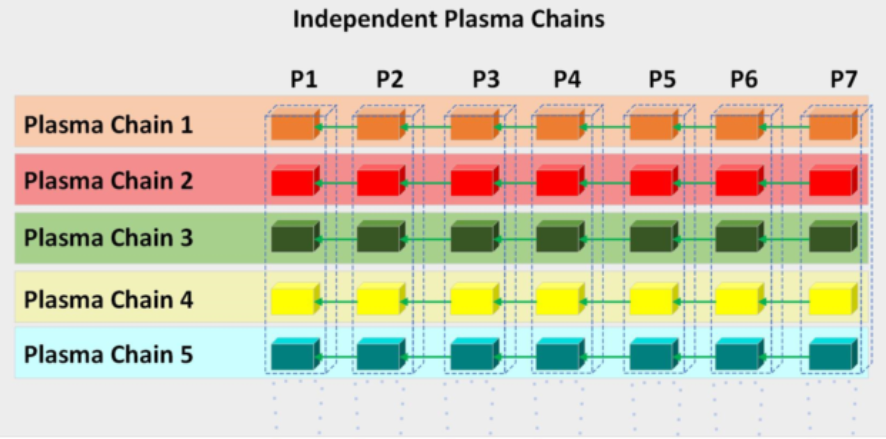
limitations.

Plasma:

1. Introduction:

Plasma is a framework proposed for encouraging the execution of smart contracts which is scalable to a larger number of states per second which essentially improves the blockchain and allows it to represent the decentralized application. The smart contracts are made to operate autonomously via transaction fees. This network transaction depends on the original Ethereum blockchain.

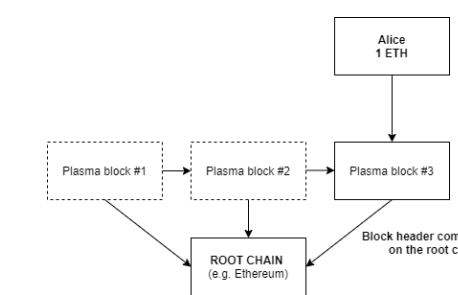
Plasma is a layer 2 solution for Ethereum and was proposed in 2017 by Joseph Poon and Vitalik Buterin. Plasma is also a sidechain solution that works separate from the original blockchain. A regular sidechain allows the user to deposit the assets and data on a contract located in the main chain which is monitored by the operator for the sidechain which then credits the assets to the users directly. There are many different mechanisms involved which allows the blockchain to experience faster speeds. This is as observed a massive improvement in the original blockchain design but this increase in speed comes at the cost of less centralization and decrease in the security which is not a favorable solution. These regular sidechains need the trust of the operator. Plasma is also based on sidechains but its aim was to solve this issue by publishing each sidechains block header to the main Ethereum chain. Even though this would minimize the trust it would at the same time allow the fraud proofs to be verified and enforces since the main Ethereum chain is very safe and always available. This also decreases the storage that is required by the main chain. This is done y compacting the state transitions in a single merkle root. There are several advantages of using plasma as multiple plasma chains can be created with a different task each. A plasma chain that only handles transfers and a plasma chain that takes care of exchange can be created simultaneously and separately. These chains then can be organized in the structure of a tree where the higher node can solve the problems faced by the lower node and evaluate and validate the proof of frauds. This structure allows the external parties or nodes to hold the funds. Since plasma runs over the root/main chain the user does not have to create transactions on the main chain for every state transition. Anyone can create a custom plasma chain for the smart contracts for various use cases. It is essentially a series of smart contracts which allows multiple chains within the main chain. This main chain is the enforcer of all the transactions. Plasma is composed of EVM smart contracts running on Ethereum.



Components:

Plasma has five main components-

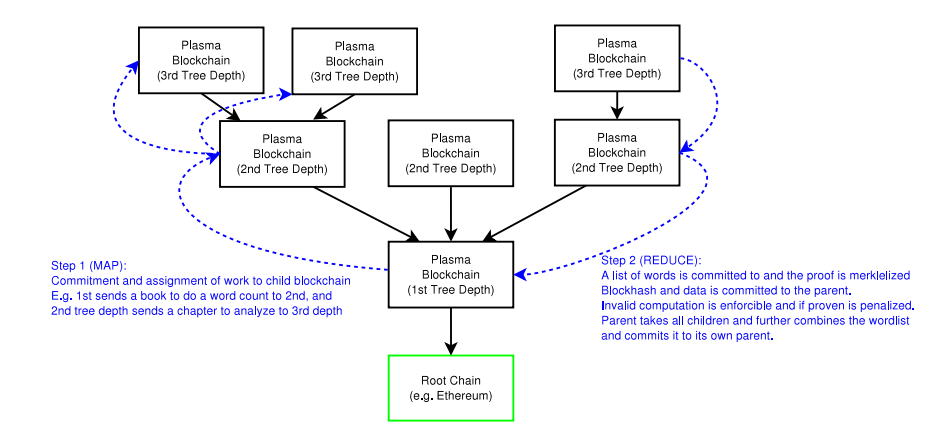
1. Incentive layer
2. Structure for child chains
3. MapReduce framework
4. Consensus mechanism
5. UTXO commitment structure



Taking the above structure as an example any participant can transfer the funds to whoever they want to and these transfers credit and withdraw the funds in the main blockchains tokens. Plasma allows the user to utilize the blockchain without the full ledger that is present in the main chain. In Plasma smart contracts are constructed as a series of fraud proofs on the main chain so that the attempts at fraud can be slashed. The side chains in plasma are operated by the validators. These validators are the ones which propose blocks and so the enforceable state restricts the malicious behaviour by making use of the fraud proofs. In a scenario where the malicious block is generated any other actor that is monitoring the sidechain who received the block can then record the merkleized fraud proof. In such a case the malicious block is then rolled back and the validator who had proposed this block is then reprimanded and penalized. The data committed by the plasma sidechain to the main chain is called the merkle root of that block which was just generated and the fraud proofs and withdrawals are secured by this merkle proof. If as a user you want to withdraw funds, your request gets submitted to the main chain contract together with the generated merkle proof which is then verified by the contract. The contract makes sure that the data output belongs to the user. A challenge period is then required to allow the other actors to provide the proof in the form of the merkle for the availability of the output. This mechanism works by using block data availability to produce the fraud proofs.

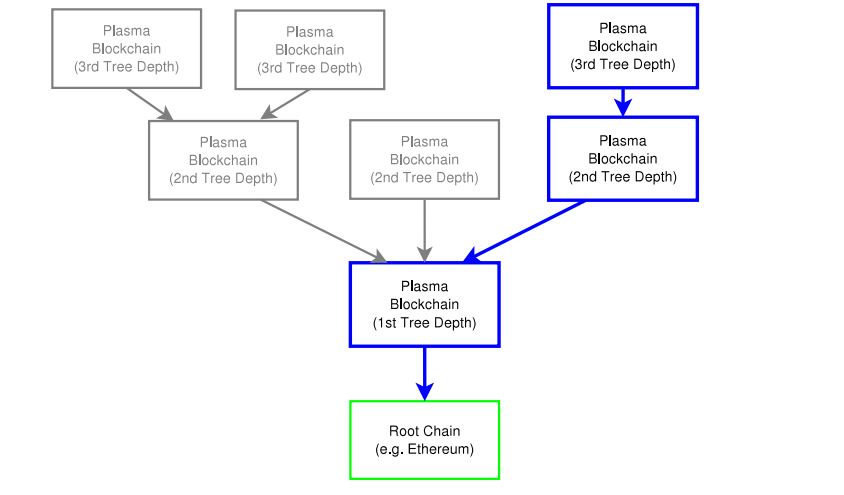
Next let us discuss about the deposits and the withdrawals. Deposits are started by the user by sending the required funds to the contract of the main chain. The plasma sidechain can the identify that the deposit has begun and include that in a block that the funds can be spent. Then the user who has deposited the funds signs a transaction on the plasma chain to declare that the he/she has approved of the commitment done by the block.

Withdrawals are comparatively slow in nature because of the additional time period required to allow other to provide fraud proofs. In this case the user sends a withdrawal transaction to the main chain. This transaction must be signed and also the amount should correspond to the unspent output. The bond amount is then put on the stake to allow for other actors to penalize for any invalid withdrawal requests. This predefined time period is the time allowed for the disputes and in a scenario a valid proof of fraud is submitted then the withdrawal is immediately cancelled and the bond is lost. This strict action is taken to make sure that additional withdrawal attempts do not take place. If this is not the case and no fraud proofs are found then the user can withdraw the funds he has requested. This may take additional time due to the disputes which though offers protection from the attempts at fraud also affect the usability and the user experience.



To take advantage of the tree structure and make the computations easier the computations are constructed in a MapReduce format. This format gives a framework high scale computation across a large number of nodes.

Blockchain: git :: Plasma: MapReduce

For examples if a message is passed down from the main block to child the child block must commit to the parent within some n number of blocks otherwise the chain would be halted. After this step the block data gives out work to the children who have given the commitment. In plasma the map phase input consists of the commitments made by the child to the parent and in the step to reduce it includes the merkleized proof of the transition while returning the result. The state transition is enforced by the fraud proofs. This allows for an increased scale in the computations. In the network the nodes do the computation while the participants verify them. The benefit is shown in scaling as the removal of the requirement to watch the chains. To reduce the affect the plasma chains can be combined together as a part of the reduce step. You do not have to care about the order but can observe all the chains as one unit where your own chain is validated. 

Scalibility

Scalibility is one of the mains reasons for the search for level 2 solutions and the plasma chain improves the scalibility achieved by the regular sidechains. The throughput and latency depend on the mechanisms thay plasma chain exploits.

Transactions per day TPD = 1500000

Blocks per day = 6,500

Average transactions per block = TPD/Blocks ~ 230

Current ethereum mainnet block gas limit = 12500000

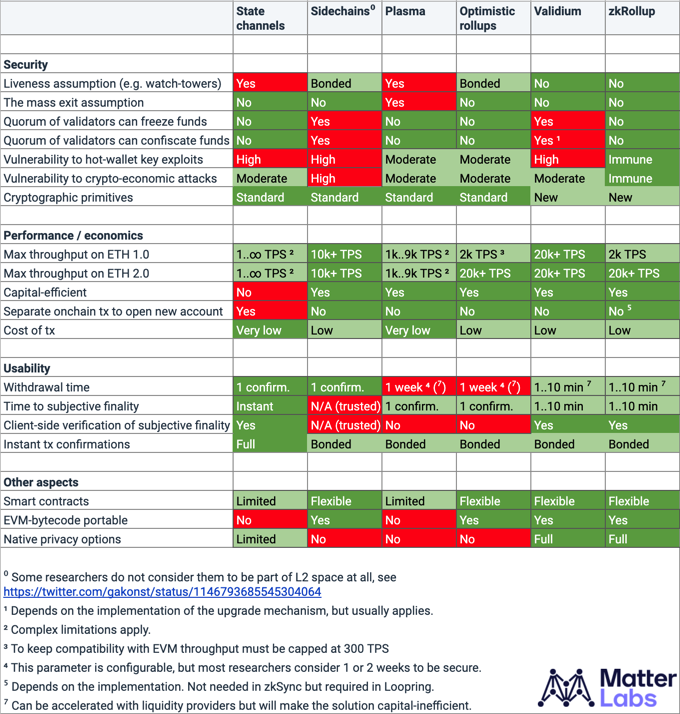
So the average gas per transaction is 12500000/230 ~ 54,350

Throughput t ≈ 20, 000, 000 Average gas per transaction · 1 2.1 seconds ≈ 175 T P S

This is a major improvement over the Ethereum throughput. This shows that the blockchains which have a similar consensus mechanism exploit more block gas limit and so can potentially lower the block time. Also, other plasma sidechains can also be added for horizontal scaling.

DECENTRALIZATION IN PLASMA

The decentralization in plasma is not a fixed quantity and it depends overall on the consensus mechanism that the chain implements. It also depends on the time when the transactions are finalized. The transaction itself is finalized only when the merkle root of the block that contains the transaction gets committed to the main chain. This is done to maintain a secure environment which is taken care of by fraud proofs and makes sure that the security is then at the same level as that of the main chain. In terms of the time period required for the commitment plasma makes use of the proof-of-stake mechanism so that there is a good balance between the decentralization and the scalability.



Code for the contract in plasma Ethereum

pragma solidity ^0.4.19;

library ArrayLib {

function remove(uint256[] storage \_array, uint256 \_value)

internal

returns (bool success)

{

int256 index = indexOf(\_array, \_value);

if (index == -1) {

return false;

}

uint256 lastElement = \_array[\_array.length - 1];

\_array[uint256(index)] = lastElement;

delete \_array[\_array.length - 1];

\_array.length -= 1;

return true;

}

function indexOf(uint256[] \_array, uint256 \_value)

internal

pure

returns(int256 index)

{

for (uint256 i = 0; i < \_array.length; i++) {

if (\_array[i] == \_value) {

return int256(i);

}

}

return -1;

}

}

pragma solidity ^0.4.17;

contract Migrations {

address public owner;

uint public last\_completed\_migration;

modifier restricted() {

if (msg.sender == owner) \_;

}

function Migrations() public {

owner = msg.sender;

}

function setCompleted(uint completed) public restricted {

last\_completed\_migration = completed;

}

function upgrade(address new\_address) public restricted {

Migrations upgraded = Migrations(new\_address);

upgraded.setCompleted(last\_completed\_migration);

}

}

pragma solidity ^0.4.19;

library MinHeapLib {

struct Heap {

uint256[] data;

}

function add(Heap storage \_heap, uint256 value) internal {

uint index = \_heap.data.length;

\_heap.data.length += 1;

\_heap.data[index] = value;

// Fix the min heap if it is violated.

while (index != 0 && \_heap.data[index] < \_heap.data[(index - 1) / 2]) {

uint256 temp = \_heap.data[index];

\_heap.data[index] = \_heap.data[(index - 1) / 2];

\_heap.data[(index - 1) / 2] = temp;

index = (index - 1) / 2;

}

}

function peek(Heap storage \_heap) view internal returns (uint256 value) {

require(\_heap.data.length > 0);

return \_heap.data[0];

}

function pop(Heap storage \_heap) internal returns (uint256 value) {

require(\_heap.data.length > 0);

uint256 root = \_heap.data[0];

\_heap.data[0] = \_heap.data[\_heap.data.length - 1];

\_heap.data.length -= 1;

heapify(\_heap, 0);

return root;

}

function heapify(Heap storage \_heap, uint i) internal {

uint left = 2 \* i + 1;

uint right = 2 \* i + 2;

uint smallest = i;

if (left < \_heap.data.length && \_heap.data[left] < \_heap.data[i]) {

smallest = left;

}

if (right < \_heap.data.length && \_heap.data[right] < \_heap.data[smallest]) {

smallest = right;

}

if (smallest != i) {

uint256 temp = \_heap.data[i];

\_heap.data[i] = \_heap.data[smallest];

\_heap.data[smallest] = temp;

heapify(\_heap, smallest);

}

}

function isEmpty(Heap storage \_heap) view internal returns (bool empty) {

return \_heap.data.length == 0;

}

}

pragma solidity ^0.4.19;

import './RLP.sol';

import './MinHeapLib.sol';

import './ArrayLib.sol';

contract PlasmaChainManager {

using ArrayLib for uint256[];

using RLP for bytes;

using RLP for RLP.RLPItem;

using RLP for RLP.Iterator;

using MinHeapLib for MinHeapLib.Heap;

bytes constant PersonalMessagePrefixBytes = "\x19Ethereum Signed Message:\n96";

uint32 constant blockHeaderLength = 161;

uint256 exitAgeOffset;

uint256 exitWaitOffset;

struct BlockHeader {

uint256 blockNumber;

bytes32 previousHash;

bytes32 merkleRoot;

bytes32 r;

bytes32 s;

uint8 v;

uint256 timeSubmitted;

}

struct DepositRecord {

uint256 blockNumber;

uint256 txIndex;

address depositor;

uint256 amount;

uint256 timeCreated;

}

struct WithdrawRecord {

uint256 blockNumber;

uint256 txIndex;

uint256 oIndex;

address beneficiary;

uint256 amount;

uint256 priority;

}

address public owner;

uint256 public lastBlockNumber;

uint256 public txCounter;

mapping(uint256 => BlockHeader) public headers;

mapping(address => DepositRecord[]) public depositRecords;

mapping(uint256 => uint256[]) public withdrawalIds;

mapping(uint256 => WithdrawRecord) public withdrawRecords;

MinHeapLib.Heap exits;

function PlasmaChainManager(uint256 exitAge, uint256 exitWait) public {

owner = msg.sender;

lastBlockNumber = 0;

txCounter = 0;

exitAgeOffset = exitAge;

exitWaitOffset = exitWait;

}

event HeaderSubmittedEvent(address signer, uint32 blockNumber);

function submitBlockHeader(bytes header) public returns (bool success) {

require(header.length == blockHeaderLength);

bytes32 blockNumber;

bytes32 previousHash;

bytes32 merkleRoot;

bytes32 sigR;

bytes32 sigS;

bytes1 sigV;

assembly {

let data := add(header, 0x20)

blockNumber := mload(data)

previousHash := mload(add(data, 32))

merkleRoot := mload(add(data, 64))

sigR := mload(add(data, 96))

sigS := mload(add(data, 128))

sigV := mload(add(data, 160))

if lt(sigV, 27) { sigV := add(sigV, 27) }

}

// Check the block number.

require(uint8(blockNumber) == lastBlockNumber + 1);

// Check the signature.

bytes32 blockHash = keccak256(PersonalMessagePrefixBytes, blockNumber,

previousHash, merkleRoot);

address signer = ecrecover(blockHash, uint8(sigV), sigR, sigS);

require(msg.sender == signer);

// Append the new header.

BlockHeader memory newHeader = BlockHeader({

blockNumber: uint8(blockNumber),

previousHash: previousHash,

merkleRoot: merkleRoot,

r: sigR,

s: sigS,

v: uint8(sigV),

timeSubmitted: now

});

headers[uint8(blockNumber)] = newHeader;

// Increment the block number by 1 and reset the transaction counter.

lastBlockNumber += 1;

txCounter = 0;

HeaderSubmittedEvent(signer, uint8(blockNumber));

return true;

}

event DepositEvent(address from, uint256 amount,

uint256 indexed blockNumber, uint256 txIndex);

function deposit() payable public returns (bool success) {

DepositRecord memory newDeposit = DepositRecord({

blockNumber: lastBlockNumber,

txIndex: txCounter,

depositor: msg.sender,

amount: msg.value,

timeCreated: now

});

depositRecords[msg.sender].push(newDeposit);

txCounter += 1;

DepositEvent(msg.sender, msg.value, newDeposit.blockNumber,

newDeposit.txIndex);

return true;

}

event WithdrawalStartedEvent(uint256 withdrawalId);

function startWithdrawal(

uint256 blockNumber,

uint256 txIndex,

uint256 oIndex,

bytes targetTx,

bytes proof

)

public

returns (uint256 withdrawalId)

{

BlockHeader memory header = headers[blockNumber];

require(header.blockNumber > 0);

var txList = targetTx.toRLPItem().toList();

require(txList.length == 13);

// Check if the target transaction is in the block.

require(isValidProof(header.merkleRoot, targetTx, proof));

// Check if the transaction owner is the sender.

address txOwner = txList[6 + 2 \* oIndex].toAddress();

require(txOwner == msg.sender);

// Generate a new withdrawal ID.

uint256 priority = max(header.timeSubmitted, now - exitAgeOffset);

withdrawalId = blockNumber \* 1000000 + txIndex \* 1000 + oIndex;

WithdrawRecord storage record = withdrawRecords[withdrawalId];

require(record.blockNumber == 0);

// Construct a new withdrawal.

record.blockNumber = blockNumber;

record.txIndex = txIndex;

record.oIndex = oIndex;

record.beneficiary = txOwner;

record.amount = txList[7 + 2 \* oIndex].toUint();

record.priority = priority;

exits.add(priority);

withdrawalIds[priority].push(withdrawalId);

WithdrawalStartedEvent(withdrawalId);

return withdrawalId;

}

event WithdrawalChallengedEvent(uint256 withdrawalId);

function challengeWithdrawal(

uint256 withdrawalId,

uint256 blockNumber,

uint256 txIndex,

uint256 oIndex,

bytes targetTx,

bytes proof

)

public

returns (bool success)

{

BlockHeader memory header = headers[blockNumber];

require(header.blockNumber > 0);

var txList = targetTx.toRLPItem().toList();

require(txList.length == 13);

// Check if the transaction is in the block.

require(isValidProof(header.merkleRoot, targetTx, proof));

// Check if the withdrawal exists.

WithdrawRecord memory record = withdrawRecords[withdrawalId];

require(record.blockNumber > 0);

// The transaction spends the given withdrawal on plasma chain.

if (isWithdrawalSpent(targetTx, record)) {

withdrawalIds[record.priority].remove(withdrawalId);

delete withdrawRecords[withdrawalId];

WithdrawalChallengedEvent(withdrawalId);

return true;

}

return false;

}

event WithdrawalCompleteEvent(uint256 indexed blockNumber,

uint256 exitBlockNumber, uint256 exitTxIndex, uint256 exitOIndex);

function finalizeWithdrawal() public returns (bool success) {

while (!exits.isEmpty() && now > exits.peek() + exitWaitOffset) {

uint256 priority = exits.pop();

for (uint256 i = 0; i < withdrawalIds[priority].length; i++) {

uint256 index = withdrawalIds[priority][i];

WithdrawRecord memory record = withdrawRecords[index];

record.beneficiary.transfer(record.amount);

WithdrawalCompleteEvent(lastBlockNumber, record.blockNumber,

record.txIndex, record.oIndex);

delete withdrawRecords[index];

}

delete withdrawalIds[priority];

}

return true;

}

function isValidProof(bytes32 root, bytes target, bytes proof)

pure

internal

returns (bool valid)

{

bytes32 hash = keccak256(target);

for (uint i = 32; i < proof.length; i += 33) {

bytes1 flag;

bytes32 sibling;

assembly {

flag := mload(add(proof, i))

sibling := mload(add(add(proof, i), 1))

}

if (flag == 0) {

hash = keccak256(sibling, hash);

} else if (flag == 1) {

hash = keccak256(hash, sibling);

}

}

return hash == root;

}

function max(uint256 a, uint256 b) pure internal returns (uint256 result) {

return (a > b) ? a : b;

}

function isWithdrawalSpent(bytes targetTx, WithdrawRecord record)

view

internal

returns (bool spent)

{

var txList = targetTx.toRLPItem().toList();

require(txList.length == 13);

// Check two inputs individually if it spent the given withdrawal.

for (uint256 i = 0; i < 2; i++) {

if (!txList[3 \* i].isEmpty()) {

uint256 blockNumber = txList[3 \* i].toUint();

// RLP will encode integer 0 to 0x80 just like empty content...

uint256 txIndex = txList[3 \* i + 1].isEmpty() ? 0 : txList[3 \* i + 1].toUint();

uint256 oIndex = txList[3 \* i + 2].isEmpty() ? 0 : txList[3 \* i + 2].toUint();

if (record.blockNumber == blockNumber &&

record.txIndex == txIndex &&

record.oIndex == oIndex) {

return true;

}

}

}

return false;

}

}

Optimistic Rollups

INTRODUCTION

Optimistic Rollup is a Layer 2 (L2) protocol designed to extend the throughput of Ethereum's base layer. Processing off-chain transactions reduces computation on the main Ethereum chain, greatly increasing processing speed. Unlike other scaling solutions like sidechains, Optimistic Rollup derives security from the mainnet by exposing transaction results on-chain or on the plasma chain. It validates transactions on Ethereum with proof of fraud, but stores the transaction data elsewhere.  
  
Computation is a time consuming and expensive part of using Ethereum, so the optimistic rollup provides a 10x to 100x scalability improvement. Optimistic rollup writes transactions to his Ethereum as call data, reducing gas costs for users.

## information

Optimistic Rollup is an approach to scaling Ethereum that involves moving computation and state storage off-chain. Optimistic Rollup executes transactions outside of Ethereum, but posts transaction data to mainnet as call data.  
  
The optimistic rollup operator groups multiple off-chain transactions into large batches before committing them to Ethereum. This approach reduces end-user fees as fixed costs can be spread across multiple transactions in each batch. Optimistic Rollup uses compression techniques to reduce the amount of data posted on Ethereum.  
  
Optimistic rollup is considered "optimistic" because it assumes that off-chain transactions are valid and does not expose proof of validity of a set of transactions posted on-chain. This distinguishes between optimistic rollups and zero-knowledge rollups that expose cryptographic verification of off-chain transactions.  
  
Optimistic Rollup relies on an incorrect scheme to detect instances of incorrectly calculated transactions. After a rollup batch is sent to his Ethereum, there is a time window (called the challenge period) during which anyone can calculate a proof of fraud and challenge the outcome of the rollup transaction.  
  
If the fraud proof is successful, the rollup log will replay the transaction and update the rollup status accordingly. Another implication of successful fraud detection is that sequencers that insert fraudulently executed transactions into blocks will be penalized.  
  
If a rollup batch has not been challenged after the challenge period has ended (i.e. all transactions have been executed correctly), it will be considered valid and accepted on Ethereum. It is possible to build on unconfirmed rollup blocks, but with one caveat. Transaction results are rolled back if they are based on a previously issued, fraudulently executed transaction.

NEED

Ethereum is arguably the most popular choice for developing decentralized applications and smart contracts.  
  
However, the ever-increasing amount of activity on Ethereum has come at a significant cost in the form of scalability. What exactly is the problem here? It raises the issue of limiting block time and block size. Limits may seem necessary to ensure security and decentralization, but limits reduce Ethereum's scalability.  
  
Ethereum experienced significant network congestion when transaction requests continuously flowed in. Users are facing slower transaction speeds and higher transaction fees. These issues could reduce the ability of the Ethereum network to accept more transactions and users. The future of the Ethereum blockchain is unimaginable without scalability. Therefore, scaling solutions such as sidechains or layer 2 rollups are required to solve such problems.

## Interaction

Optimistic Rollup is an off-chain scaling solution designed to run on Ethereum. Each optimistic rollup is governed by a set of smart contracts deployed on the Ethereum network. Optimistic rollup processes transactions outside of the Ethereum main chain, but registers off-chain transactions (in batches) into an on-chain rollup contract. Similar to the Ethereum blockchain, this transaction record is immutable, forming an optimistic rollup chain.

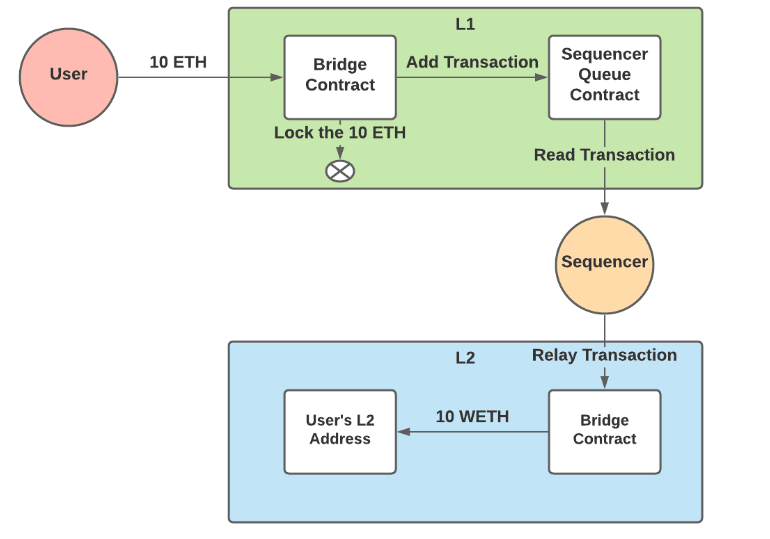
ARCHITECTURE

The architecture of an optimistic rollup comprises of the following parts:

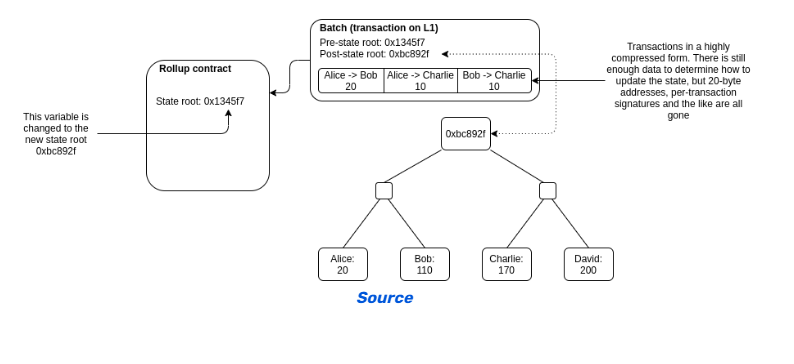
On-chain contracts: Optimistic rollup operations are controlled by smart contracts running on Ethereum. This includes contracts that store rollup blocks, monitor rollup status updates, and track user deposits. In this sense, Ethereum acts as a base layer or “Layer 1” for an optimistic rollup.

Off-chain virtual machine (VM): The contract governing the optimistic rollup protocol runs on Ethereum, but the rollup protocol performs computations and stores state in a virtual machine separate from the Ethereum virtual machine. Off-chain VMs are where applications run and state changes are performed. Serves as the top layer or "Layer 2" for optimistic rollups.  
  
Optimistic Rollup is designed to run programs written or compiled for EVM, so off-chain VMs contain many EVM design specs. Additionally, proof of fraud computed on-chain allows the Ethereum network to enforce the validity of state changes computed on off-chain VMs.

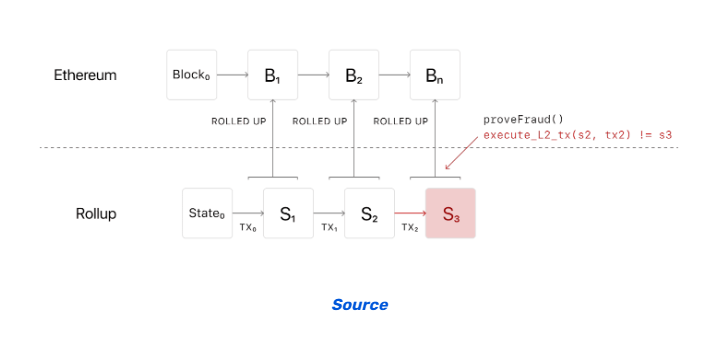
#### 1. Entering the optimistic rollup



#### 2. Using the optimistic rollup

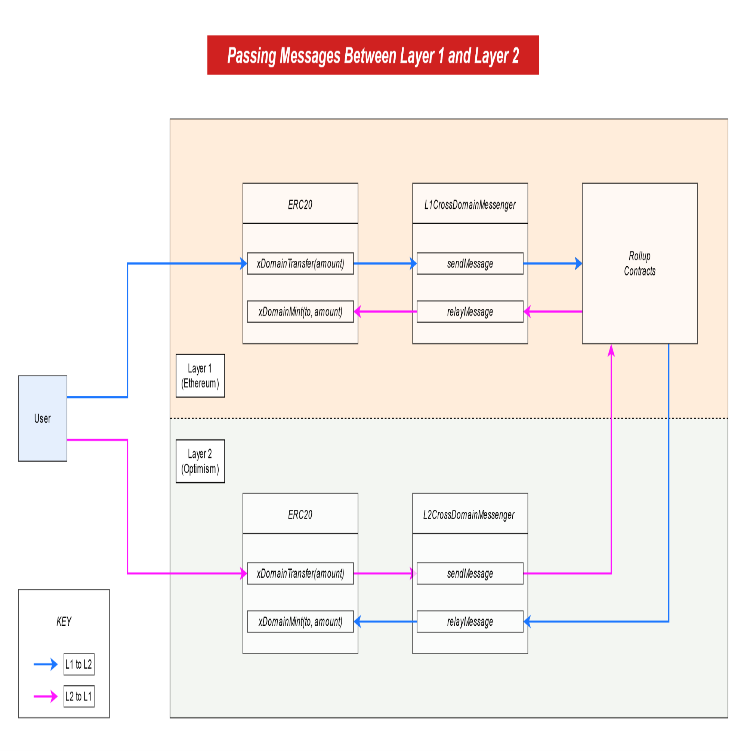


#### 3. Exiting the optimistic rollup



### WORKING OF OPTIMISTIC ROLLUPS

The mechanism that defines the function of optimistic rollup is just a glimpse of its power. On the contrary, you should look for a comprehensive explanation of how it works. Here, we describe in detail how optimistic rollup works and examine the different stages of how it works.



#### Executing and Aggregating Transactions

Operators are responsible for process-optimized rollup transactions. Users must submit transactions to operators, also known as aggregators or validators. The validator works on aggregating transactions, compressing the underlying data, and publishing the block in question on her Ethereum.  
  
It's also important to note that any user can be responsible for validators. However, optimistic rollup validators also need to provide bindings before creating blocks, similar to proof-of-stake systems. Validators should penalize bonds for publishing invalid blocks or building old invalid blocks.  
  
Other validators in the optimistic rollup example should use a single copy of the rollup state to guarantee transaction execution. If the operator deviates from the proposed state to the validator's final state, the validator can dispute the result and compute fraud protection.  
  
It is also important to note that certain optimistic rollups can bypass unauthorized verification systems. Such a rollup relies on a single "sequencer" to run the chain, assuming all the responsibilities of the validator. However, the sequencer provides better control over transaction order and priority access to rollup chains. Furthermore, the sequencer also has sole authority to submit transactions to the on-chain contract.

#### Sending the Rollup Blocks to Ethereum

The operator aggregates various off-chain transactions into one bundle and sends it to Ethereum. This process using optimistic rollup must address transactional data compression. I should also mention that Ethereum requires transaction data to be published in the form of “call data”.

"Call data" is the non-persistent, immutable highlight of smart contracts, and acts primarily like memory. It should be noted that the call data may remain in the on-chain blockchain history log even if it was not found in the Ethereum state. We can see that our optimistic rollup can reduce on-chain data storage costs.  
  
It should also be noted that Solidity's 'calldata' keyword helps pass arguments to smart contract functions at runtime. It works to identify the functions called in a transaction and sort the inputs in any order. The answer to the question "How does optimistic rollup work?" also draws attention to other meanings of "call day". These are very useful elements for sending compressed transaction data to on-chain contracts.

#### State Commitments

The next culmination of understanding “what is an optimistic rollup” is to highlight government commitment. First, note that optimistic rollup states are always organized in the form of a Merkle tree. A Merkle tree is also called a "state tree" that contains all the information about the rollup state. The root of the state tree points to the final state of the rollup, hashed and stored in the rollup contract. Every state transition in a chain can produce a new rollup state that is committed by an operator that computes a new state root.  
  
Operators should submit old and new status roots at batch release. The rollup operator also needs to ensure that the transaction stack has a Merkle root, so that it can prove that the layer 1 stack contains the transaction. State commitment is a key highlight of the Optimistic Rollup project because it allows you to prove the correctness of Optimistic Rollup's state changes.

#### Fraud Proof

The most important aspect of using optimistic rollup is the emphasis on fraud prevention. Optimistic Rollup can help everyone with a public block without entering verification. However, users can use optimistic rollups to challenge state transitions within a specific timeframe. If the rollup block is challenged, the protocol begins anti-fraud calculations. There are many variations on the number of interactions required to perform anti-fraud calculations.  
  
For single-round interactive validation schemes, conflicting transactions can be repeated in the layer 1 network to detect invalid blocks. A detailed understanding of how optimistic rollup works also focuses on emulating the replay of disputed transactions at Layer 1 using validator contracts. If the challenger turns out to be correct, the operator will have to bear the penalty with the security deposit.  
  
However, it is important to note that replaying transactions at layer 1 has some important requirements. In addition to government mandates, rollups require more data to be published on-chain for each individual transaction, incurring gas costs to replay the transaction. Therefore, the multi-round interactive checking mechanism increases efficiency.

#### Multi-round Interactive Proving

The interactive multi-round proof element is another important aspect of understanding the optimistic rollup example and its functionality. Multi-round interactive proofs fundamentally rely on interactions between the challenger and the claimant or operator. Interactions are subject to oversight of the Layer 1 Verifier Agreement that determines the veracity of the operator's claims.  
  
After querying the rollup block, the operator should split the assertion in question into her two equal parts. All rollup blocks have similar computation steps. Then, when the challenger chooses the rollup block, the dichotomous protocol begins the splitting process and continues until both parties address the dispute over the rollup block. In such cases, layer 1 contracts work to resolve disputes by identifying the misbehaving party.

The Fraud Check is a distinct highlight in understanding how optimistic rollups work in ensuring reliable finality. An optimistic rollup can easily guarantee that a transaction is valid and can finally be confirmed. Evidence of fraud can ultimately prove the effectiveness of rollup blocking. As a result, optimistic rollups are more likely to penalize malicious nodes that challenge honest operators.

PROS and CONS

Pros:

* Flexibility in generalized Computation (Turing-complete / EVM compatible)
* Increase in scalability (200 to 2000 transactions-per-second (tps) vs Ethereum layer 1's current 10 tps)
* All Data is available on-chain (no need to trust off-chain data providers)
* Better UX (as explained above)

Cons:

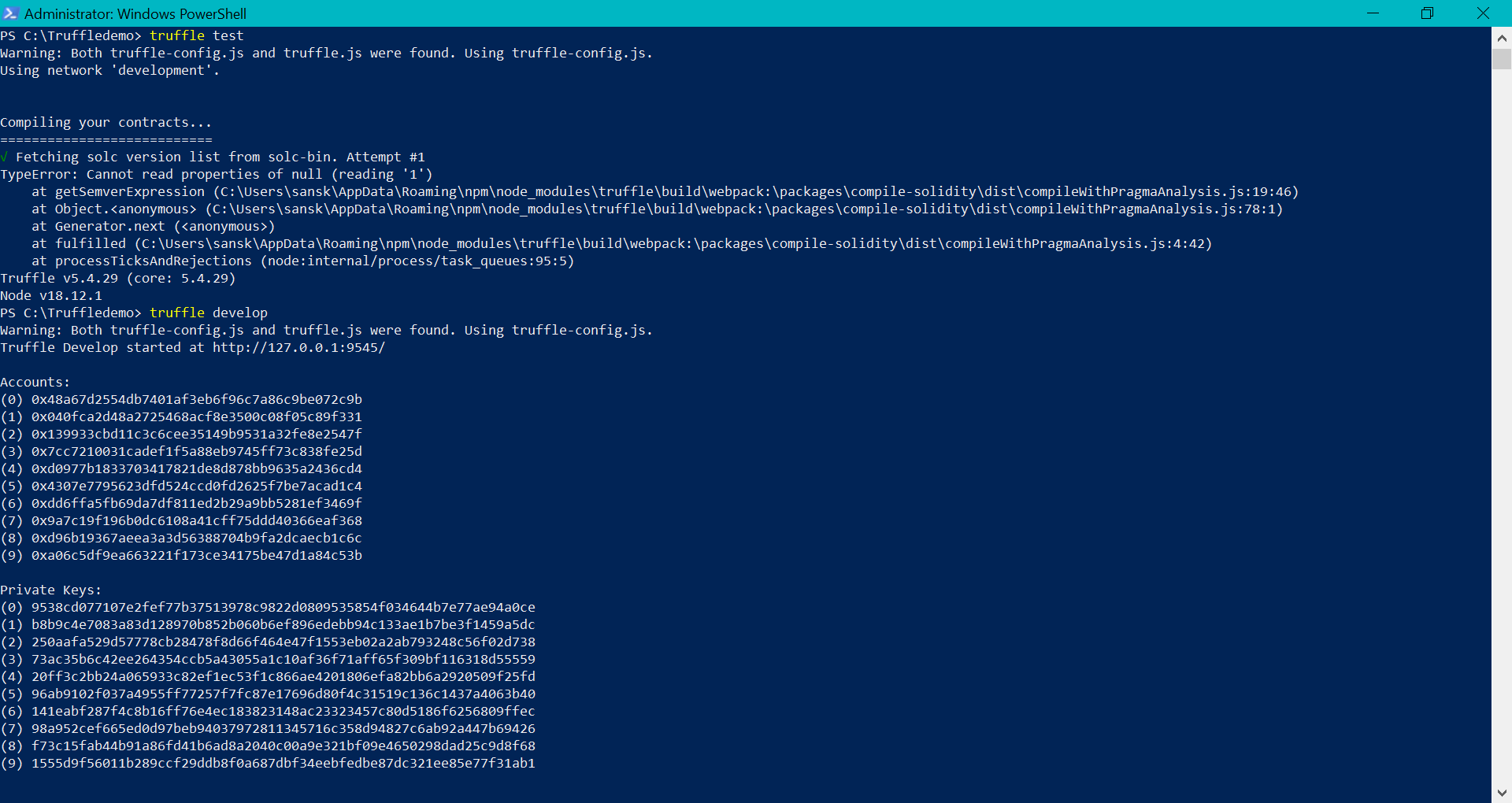
* Limited Throughput when compared with some other Layer 2 solutions (Plasma, ZK Rollups, etc.)
* Some additional security issues are raised

CONCLUSION

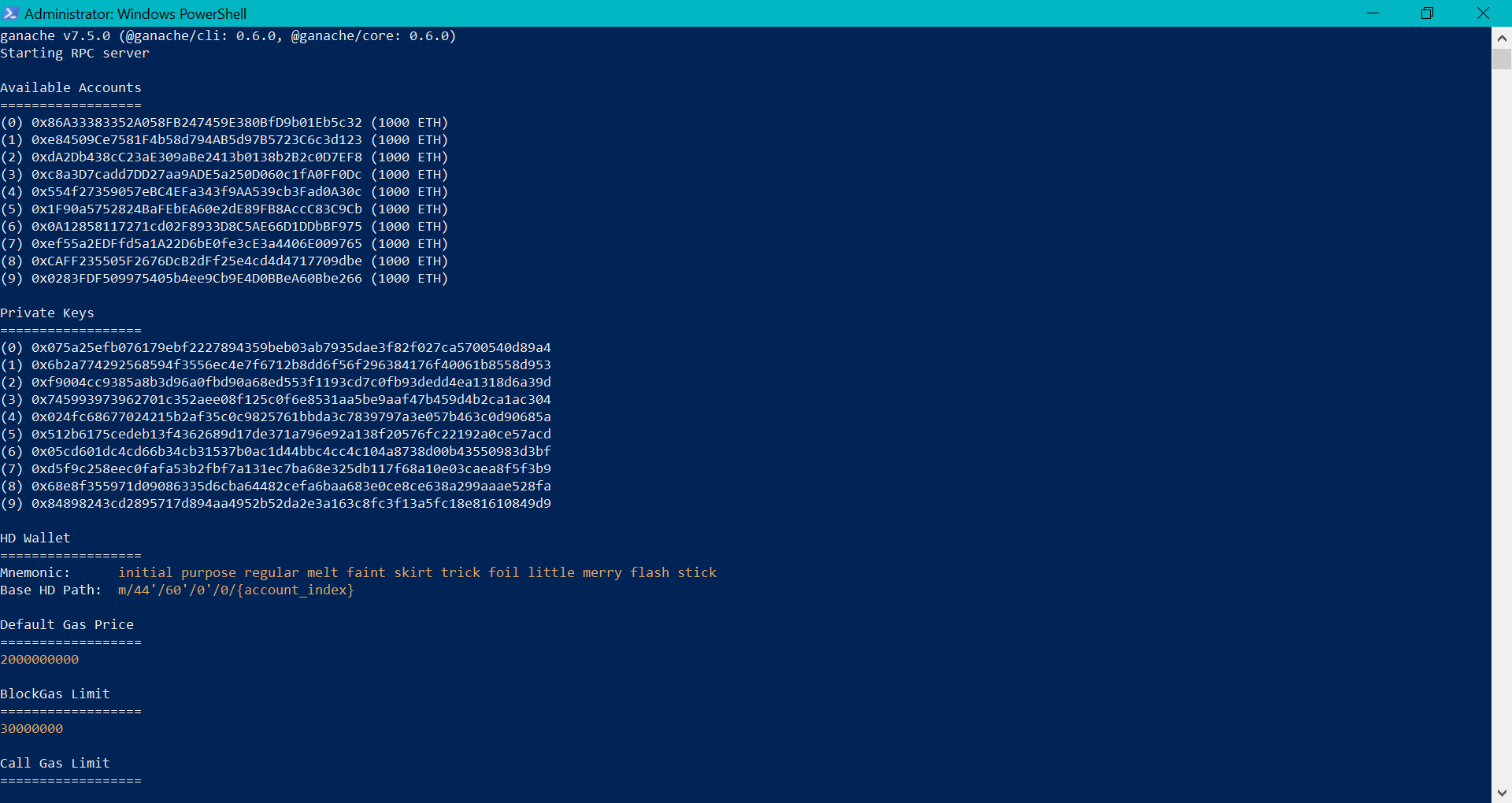
An important highlight in the discussion of optimistic rollup would be to draw attention to its strengths and weaknesses. We can provide. Storing transactional data at Layer 1 improves decentralization, security, and censorship resistance. Additionally, Optimistic Rollup's compatibility with Solidity and EVM allows porting Ethereum-based smart contracts to Rollup.  
  
However, optimistic rollup also introduces some obstacles, such as delayed transaction finality and possible power abuse by the sequencer. Additionally, the lack of honest nodes can increase the likelihood of fraud due to invalid government obligations and bans. may be connected.

OUTPUT SCREENSHOTS

TRUFFLE IDE



GANACHE



.

##### References

1.Özhan Sağlık (2022) Searching for trust: blockchain technology in an age of disinformation, Archives and Records, 43:3, 340-342, DOI: [10.1080/23257962.2022.2108010](https://doi.org/10.1080/23257962.2022.2108010)

2.Weichao Gao, William G. Hatcher, and Wei Yu. 2018. A survey of blockchain: Techniques, applications, and challenges. In the 27th International Conference on Computer Communication and Network. 1—11

3.Satoshi Nakamoto. 2019. Bitcoin: A peer-to-peer electronic cash system. Retrieved from <http://bitcoin.org/bitcoin.pdf>.

4.Losbar, Yannick. "How Can the Bitcoin Blockchain Help Humanity?" CryptoCoinsNews. 5.CryptoCoinsNews, 16 Sept. 2014. Web. 02 Jan. 2015

Twesige, Richard L. E-FREIGHT Implementation of Electronic Documents in a Paper-based Environment. 6.N.p.: Reasearchgate, 29 Dec. 2014. PDF.

Twesige, R. L. (2015). A simple explanation of Bitcoin and Blockchain technology.

7.Bischoff, O., & Seuring, S. (2021). Opportunities and limitations of public blockchain-based supply chain traceability. *Modern Supply Chain Research and Applications*.

8.Abeyratne, S.A. and Monfared, R.P. (2016), “*Blockchain ready manufacturing supply chain using distributed ledger*”, International Journal of Research in Engineering and Technology, Vol. 5 No. 9, pp. 1-10.

9.Babich, V. and Hilary, G. (2020), “*OM forum—distributed ledgers and operations: what operations management researchers should know about blockchain technology*”, Manufacturing and Service Operations Management, Vol. 22 No. 2, pp. 223-240.

10.Bai, C. and Sarkis, J. (2020), “*A supply chain transparency and sustainability technology appraisal model for blockchain technology*”, International Journal of Production Research, Vol. 58 No. 7, pp. 2142-2162.

11.Hughes, L., Dwivedi, Y. K., Misra, S. K., Rana, N. P., Raghavan, V., & Akella, V. (2019). Blockchain research, practice and policy: Applications, benefits, limitations, emerging research themes and research agenda. *International Journal of Information Management*, *49*, 114-129.

12.Sharma, S., Rosmin, P., & Bhagat, A. (2021). Blockchain Technology: Limitations and Future Possibilities. In *Blockchain Applications in IoT Security* (pp. 140-151). IGI Global.

13.Golosova, J., & Romanovs, A. (2018, November). The advantages and disadvantages of the blockchain technology. In *2018 IEEE 6th workshop on advances in information, electronic and electrical engineering (AIEEE)* (pp. 1-6). IEEE.

14.A. Songara and L. Chouhan, "Blockchain: A Decentralized Technique for Securing Internet of Things", Conference paper, October 2017.

Advantages and disadvantages of Blockchain Technology, 2018, [online] Available: <https://dataflair.training/blogs/advantages-and-disadvantages-of-blockchain/>.

15.Sguanci, C., Spatafora, R., & Vergani, A. M. (2021). Layer 2 blockchain scaling: A survey. *arXiv preprint arXiv:2107.10881*.

16.Schaffner, T. (2021). Scaling Public Blockchains. *A comprehensive analysis of optimistic and zero-knowledge rollups. University of Basel*.

17.Cherukupally, S. (2021). Blockchain technology: Theory and practice. In *Handbook of statistics* (Vol. 44, pp. 75-103). Elsevier.

18.1Gangwal, A., Gangavalli, H. R., & Thirupathi, A. (2022). A survey of Layer-two blockchain protocols. *Journal of Network and Computer Applications*, 103539.

9.Gudgeon, L., Moreno-Sanchez, P., Roos, S., McCorry, P., & Gervais, A. (2020, February). Sok: Layer-two blockchain protocols. In *International Conference on Financial Cryptography and Data Security* (pp. 201-226). Springer, Cham.

20.Greenberg, A., Lahiri, P., Maltz, D. A., Patel, P., & Sengupta, S. (2008, August). Towards a next generation data center architecture: scalability and commoditization. In *Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow* (pp. 57-62).

21.2Bez, M., Fornari, G., & Vardanega, T. (2019, April). The scalability challenge of ethereum: An initial quantitative analysis. In *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)* (pp. 167-176). IEEE.

22.Poon, J., & Buterin, V. (2017). Plasma: Scalable autonomous smart contracts. *White paper*, 1-47.

Stark, J. (2018). Making sense of ethereum’s layer 2 scaling solutions: State channels, plasma, and truebit. *Medium. com*.

23.Bez, M., Fornari, G., & Vardanega, T. (2019, April). The scalability challenge of ethereum: An initial quantitative analysis. In *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)* (pp. 167-176). IEEE.

24.Poon, J., & Dryja, T. (2016). The bitcoin lightning network: Scalable off-chain instant payments.

Antonopoulos, A. M., Osuntokun, O., & Pickhardt, R. (2021). *Mastering the Lightning Network*. " O'Reilly Media, Inc.".

25.Seres, I. A., Gulyás, L., Nagy, D. A., & Burcsi, P. (2020). Topological analysis of bitcoin’s lightning network. In *Mathematical Research for Blockchain Economy* (pp. 1-12). Springer, Cham.

26.Donno, L. (2022). Optimistic and Validity Rollups: Analysis and Comparison between Optimism and StarkNet. *arXiv preprint arXiv:2210.16610*.

27.Yu, T., Luo, F., Pu, C., Zhao, Z., & Ranzi, G. (2022). Dual‐blockchain‐based P2P energy trading system with an improved optimistic rollup mechanism. *IET Smart Grid*.