Ethereum, Smart Contracts and the Optimistic Roll-up

Matthew Armstrong

A Final Year Project

Presented to the University of Dublin, Trinity College

in partial fulfilment of the requirements for the degree of

BA Mod. Computer Science

Supervisor: Donal O'Mahony

August 2021

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Matthew Armstrong

August 16, 2021

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Matthew Armstrong

Matthew Armstrong

August 16, 2021

Ethereum, Smart Contracts and the Optimistic Roll-up

Matthew Armstrong, BA Mod. Computer Science

University of Dublin, Trinity College, 2021

Supervisor: Donal O'Mahony

Abstract: The Ethereum Blockchain is the second most popular platform in the blockchain technology sector. Allowing for the use of smart contracts, which in turn ensures computational versatility on its platforms, it gives rise to a plethora of "DApp's" and Currency platforms. However, major issues with Ethereum is its latency and delay in transaction speed, as well as the high occurring costs, or "Gas" fees, to send currency from one address to another. In this project we will explore the efficiency of the "Optimistic" layer two "roll-up" in solving this Layer One transaction cost issue. Additionally, A demonstration of the improved scalability brought about by the developments in Layer 2 solutions in the Blockchain and Ethereum Eco-system will be provided.

Keywords: Layer two, Ethereum, Rollups, Scalability, Blockchain, Optimism

Acknowledgments

A large thank you to my supervisor , Dr. Donal O'Mahony for taking time every week to help and advise on the progress of this project, giving me the project in a very difficult personal time when finding someone willing to help was few and far between.

Secondly, Id like to thank my tutor Arthur White for working with me to find the resources and time required to do this project in a very trying time.

Thirdly I'd like to thank Stefan Weber for not wringing my neck for all the headaches I must have given the Final year Project registration system

Lastly I'd like to thank the gremlins of the glass rooms , the boys in the background and my best friends for literally forcing me to the finish line. you know who you are. Thank you.

Matthew Armstrong

University of Dublin, Trinity College
August 2021

Contents

List of Figures

Chapter 1

Introduction

With the rise of Cryptocurrencies and Blockchain technology within the time frame of 2009 to present day we have seen both amazing efficiencies and applications of the technology as well as deficiencies and exploitation occur. This has given rise to new areas of research within the fields of Cryptography, Security infrastructure, peer-to-peer information networks and scalability improvements within the technology itself. Blockchain technology is perceived by many as a possible solution to many real world issues currently due to its decentralised nature and inherent security and safety models. However the issue that this technology faces, before many of these goals can be achieved, is the scalability issue. As the technology stands right now it is not possible for mass adoption as simply put no Blockchain can handle the world at scale currently. With issues ranging from low transactions speeds, Exuberant computational costs, real world currency transaction costs having periodic exponential increases as well as the technology itself not being user friendly or having a low barrier to entry the Blockchain world has made rapid improvements to meet the demand for scale with varying results.

Within this document the areas of cryptography, blockchain and layer two solutions will be explored in fine detail with the aim of exploring the benefits and drawbacks of a layer two solutions, with emphasis on the "Optimism" network, in a real setting using an assortment of development technologies.

1.1 The Project Aim

The aim of this project is to identify the key issues facing Blockchain technology, specifically Ethereum, and investigate the various Layer Two solutions proposed to address these problems. In order to achieve we also need to understand, at a high level, the key components of the Blockchain ecosystem. For each of these areas, I have outlined the basic level of investigation required below:

Blockchain: Identify and understand the key components and basic functionality of Blockchain technology

Ethereum: Gain an in-depth understanding of Ethereum as a product, as well as an understanding of the current issues facing ETH 1.0. Further investigate the chain of issues that ultimately led to the development of layer two solutions

Layer Two Solutions: Explore the rapidly expanding and developing area of layer two solutions with the aim of implementing and presenting real world evidence to attest to their viability as ideal solutions. Simultaneously, analyse previous unsuccessful models to identify where and why they failed to provide resolution.

Rollups: Gather data on the current implementations of rollups being implemented on ETH 1.0 - their current capabilities and their future possibilities

Optimism: Work with the Optimism network in an attempt to verify the theoretical transactions improvements to both transaction processing and speed.

Chapter 2

State of the Art

In this chapter we will explore the State of the Art relevant to Blockchain technology namely, Ethereum, Layer two solutions, the concept of Rollups and specifically, one known as an "optimistic" rollup. The purpose of this being to examine the initial problem of "gas" fees within the Ethereum Blockchain and analyse other potential solutions that exist within the sector currently, with the aim of thoroughly examining the inner working of an Optimistic layer two transaction solution to confirm whether it is indeed the optimal solution at present. A full breakdown of all the technologies required for full understanding of both the problem and the potential solution are documented.

Blockchain technology being the foundation on which the project is based , the advancement made by the Ethereum developers upon the original infrastructure proposed in the paper "Bitcoin - A Peer to Peer Electronic Cash System" written by Satoshi Nakamoto is thoroughly researched as to have the best understanding of the high level workings of the foundation of the problem at hand.

Also required for full understanding is insight into Ethereum and its advancements upon the base idea of a Blockchain which by extension shows the current transactions issue relating to exuberant "Gas" fees caused by Ethereum's limited ability to handle large volumes of transactions on a second by second basis, which in turn brings forth the issue to be solved by Layer Two Solutions.

Various Layer Two solutions such as a channels, Plasma ,sidechains and Rollups were analyzed as many, if not all borrow concepts and technology from each other yet provide different benefits and drawbacks based on the route taken forward by each development team.

Optimistic Roll ups and they're validity as the most optimal solution to the Gas Fee issue currently affecting the Ethereum Eco-system. A dive into its improvements over past solutions such as "Plasma" to highlight the issues that are both solved and are still being

addressed. A comparison between it and ZK-Rollups, the alternative rollup technology also showing promise in the area and a comparative in real world application use and development.

A glimpse at the current working development abilities of both ZK-rollups and Optimistic rollups. A practical demonstration of the real world usefulness of the rollups as they stand in their current state.

2.1 Cryptography

The basis of Cryptography is easiest explained through the use of our three characters Alice, Bob and Oscar. Cryptography is a method by which Alice and Bob can communicate over an insecure data medium or site without the opponent, Oscar, being able to read the data being transferred between them. Encryption is a method of converting plain text and simple data into a unreadable form of text or data through the use of cryptography. This This string can be reverted into the original readable format with the correct credentials known only to the sender and the recipient, (decryption) thus ensuring giving the communication remains private and secure. This method is used for many things including secure data storage or end-to-end messaging which is only readable by the users involved and no other. In simple terms cryptography is a method in which to ensure that only the intended recipients can access the data sent to them.

In many forms of cryptography the idea of keys is used. In an example where two users, Ashley and Bradley, are sending a message, each user would have both a private and public key. The public key, as the name suggests, is obtainable by anyone not only Ashley and Bradley. However the private keys of both parties is only known to themselves. When a message is to be sent Ashley will generate a new key using both her own private key and Bradley's public key. The message sent is encrypted using this key generated by the private-public key pair. Upon receiving the message Bradley creates a key-pair using his own private key and Ashley's public key and using this key-pair he is able to decrypt and view the original message sent by Ashley.

In figure 2.1 a set of Mathematical formulae describe the definition of a *Cryptosystem*.

Definition 1.1: A *cryptosystem* is a five-tuple $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$, where the following conditions are satisfied:

1. $\mathcal{P}$ is a finite set of possible *plaintexts*;
2. $\mathcal{C}$ is a finite set of possible *ciphertexts*;
3. $\mathcal{K}$, the *keyspace*, is a finite set of possible *keys*;
4. For each $K \in \mathcal{K}$, there is an *encryption rule* $e_K \in \mathcal{E}$ and a corresponding *decryption rule* $d_K \in \mathcal{D}$. Each $e_K : \mathcal{P} \to \mathcal{C}$ and $d_K : \mathcal{C} \to \mathcal{P}$ are functions such that $d_K(e_K(x)) = x$ for every plaintext element $x \in \mathcal{P}$.

Figure 2.1: A set of Mathematical formulae describe the definition of a *Cryptosystem*

In this way cryptography is used for data security and privacy. An issue with cryptographic methods is a trust factor. The generated private-public key from Ashley might be tampered with by a third party at some point within the transaction of data. It is not possible to guarantee that the key pair generated at any point is protected from malicious changes. Therefore a trust party is required to guarantee that the public-private key pair generated is valid and authentic. There are two methods used to prove or authenticate the key-pair generated:

A public Key Infrastructure in which a third party validates the public key pairs of all users in a network and determines if malicious changes have been made at any stage in each transaction. As this validation is carried out on a third parties hardware, there must be an implicit trust between the parties involved and the third party maintaining security. An example given in figure 2.2

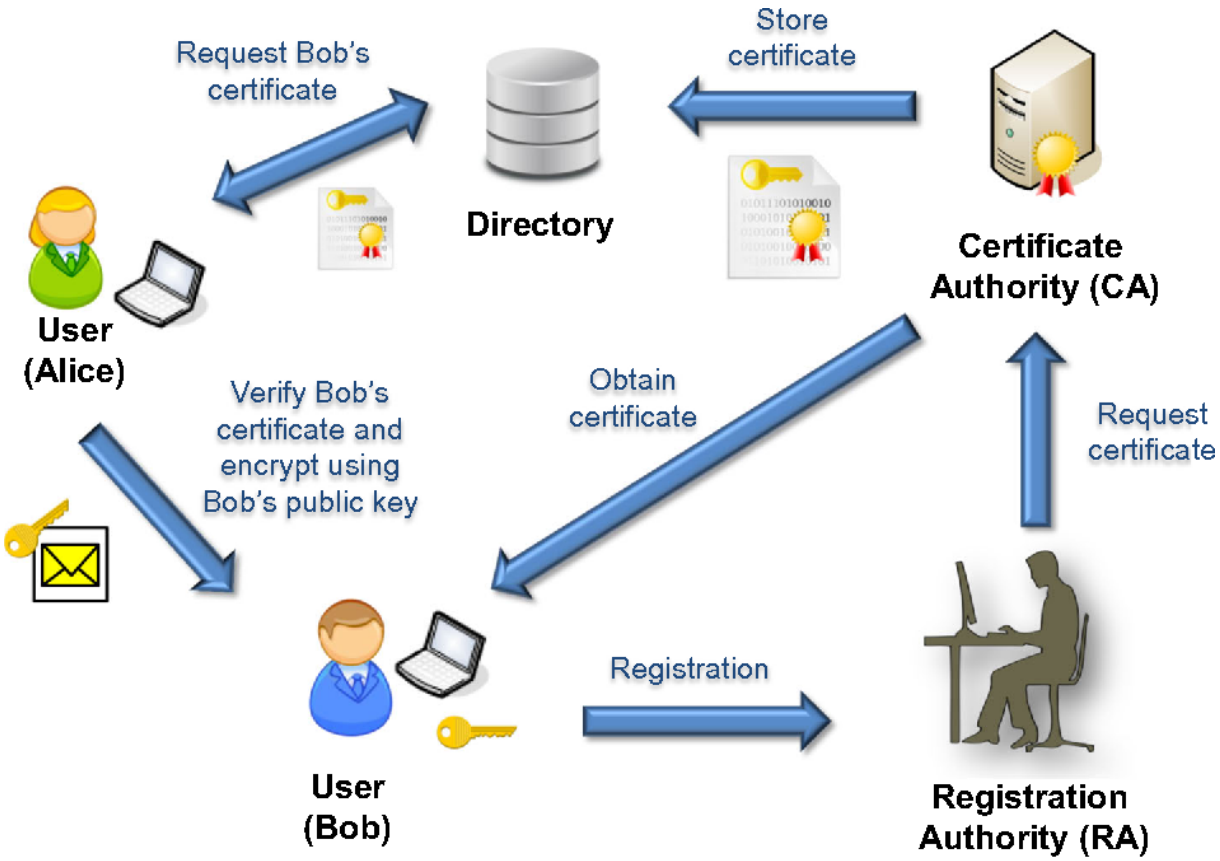

Figure 2.2: Example diagram a Public key Infrastructure transaction occurring

A “web of trust” is a decentralised method where each user has a public and private key. A key binding is made to link that public key to a specific user ensuring that any recipient can confirm that the sender and its data is indeed from the specified user. the sender then encrypts their data with the recipients public key ensuring that only the recipient can decrpyt the data sent. Other recipients validate the trust of this transaction by signing the transaction with their private key. A users trust is validated by other users around them in what is called a “ring”. This ring of users validate that a sender can be trusted with users closest to the sender confirming the level of trust considered completely valid and users further from the sender having a much less trusted confirmation of the senders validity. An example given in figure 2.3

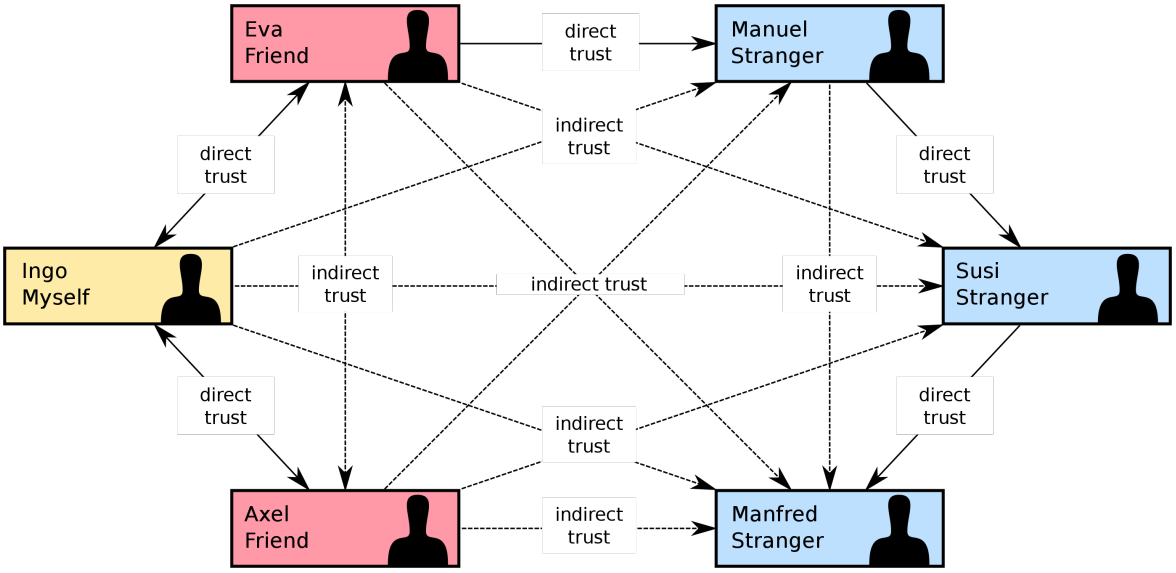

Figure 2.3: A decentralised method for certifying and validating the trustworthiness of a user and their credentials in a given transaction. The users closest to the sender have a full proof trust of the sender whereas users further away would trust the sender less as more ”hops” are required to find a user to validate this transaction

For Blockchain technology a Secure Hashing Algorithm(SHA) is used when creating the hash of each block in the form of a cryptographic proof. In Bitcoin’s case the method is SHA-256, whereas in Ethereum’s case it uses Keccak-256 which is a hash function within the SHA-3 grouping.

2.2 Blockchain Technology

Blockchain technology was original conceptualised in the paper "Bitcoin - A Peer to Peer Electronic Cash System" written by Satoshi Nakamoto. Satoshi has not been confirmed to be a single person and it is speculated that it is a name agreed upon by a collective group that worked on the original Bitcoin whitepaper. The Bitcoin whitepaper introduced a new way in which transactions could be handled without the need for a "middle man" or a financial institution for liquidity purposes. Timestamps on transactions allowed blockchain to address the "double-spend" issue, an issue in which a user would spend a specific amount of their funds in multiple places or transactions at the same time, that would usually require these third parties to oversee. The proposed technology would add benefits such as instant transactions, a drop in transaction costs and open access at all times to the public. Often referred to as a public distributed ledger, Blockchain technology is a method in which transaction details are stored in "blocks" which are appended to the previous block in a non-editable and unchangeable chain of information, known as immutable, that is checked for validity by cryptographic proofs. "We define an electronic coin as a chain of digital signatures" [18], These digital signatures are used to verify the previous owners, parties involved and specific transactions of a coin on the blockchain.

Each block in the chain contains four values:

- The Hash Value of the previous block from which the current block's hash value is generated. This guarantee's that the block is valid, If some block were to be tampered with , its hash value would change therefore invalidating all blocks ahead of it as all blocks would need their own hash values recalculated based on the change to the initial block, this being compared to the validated chain that is distributed publicly throughout the network causes the block to be invalidated and rejected from the chain restoring it to its previously healthy state.

- Data pertaining to the actions facilitated on the chain. This data can take many forms but in the simplest example , for instance Bitcoin, it is the details of a financial transaction that occurred between two parties, both of who's hash signature is recorded, as well as the value of Bitcoin that was exchanged within the transaction.

- A Nonce value, which is a randomly generated value used to add variation to the hash value of the next block when it is being generated.

- After the nonce value has been found and the block validated , the new hash value of the block will be generated with the nonce value and the previous hash giving a new unique identifiable hash value for this specific block.

2.3 Ethereum

Ethereum is the second most known blockchain technology and leader in DApp development and deployment in today's world. Inspired by the growth of Bitcoin it set out to improve on the structure on which an application could be built using its blockchain technology. Maintaining the functions of a distributed ledger , trust-less transactions and peer-to-peer networking Ethereum is set apart from Bitcoin by key differences and improvements. Ethereum uses the Keccak-256 hash function as its cryptographic method as apposed to the SHA-256 used by Bitcoin. This Keccak-256 method is the winner of a cryptographic method competition that spanned many years and was designed to be the successor of SHA-2, which contains the SHA-256 method used by Bitcoin. Soon after the development of Ethereum started the Keccak-256 method would be absorbed into the SHA-3 hash functions grouping.

Ethereum, like Bitcoin, is a Proof-Of-Work blockchain meaning that "miners" are rewarded for validating transactions on nodes with the cryptocurrency 'Ether' upon completion of a block being mined. This means that miners are competing on computationally intensive puzzles, or more precisely, trying to find a nonce that meets a specific target difficulty, which is a time and energy consuming task ([22]). Each block completed rewards one Ether, which at the time of writing has fluctuated between a worth of Two thousand to Four thousand United States dollars. Today Bitcoin is usually mined on Application-specific integrated circuit(ASIC) miners while Ethereum is usually mined on Graphics Processing Units(GPU's) found in high end Computers or Gaming Computers. This is due to the fact that Ethereum's Keccak-256 hash function is an ASIC resistant hash algorithm meaning that the architecture of a GPU, mainly its thousands of 'Cuda-Cores' makes them highly efficient at mining the EthHash algorithm.

2.3.1 The Block

As Ethereum grew and built upon previous Blockchain technology and concepts, the amount of data required for functioning block creation increased as more data per block was required to uphold the working functions of the Ethereum chain. In the original proposed yellow paper for the Ethereum Virtual Machine the parameters of the new block structure for the chain are shown. Along with the initial values described in section 2.2,

multiple other data objects are required to hold all pertinent data to be used in function calls to the chain. Along with the original data stated, the objects required for a typical Ethereum block are as follows:

- The **ParentHash** being the cryptographic hash of the parent block to the current block
- The **ommersHash** being the hash of a sibling block to the parent block.
- The **beneficiary** which is a 160-bit address that all fess gathered from the mining of the current block will be sent to.
- The **stateRoot** contains , after all transactions are finalised, the hash of the root node of the state trie.
- The **transactionsRoot** contains the hash of the root node of the state trie structure after being populated with all transactions within the block.
- The **receiptsRoot** contains the hash of the root node of the state trie structure after being populated with all receipts within the block.
- The **logsBloom** A filter composed of logger address and topics from the logs of each receipt.
- The **difficulty** value being the scalar value representing the mining computational power required to mine the current block.
- The **number** being a scalar value equivalent to the number of block prior to the current block.
- The **gasLimit** value depicts the current limit for gas expenditure for each transaction stored within the current block.
- The **gasUsed** for all transactions within the current block.
- The **timestamp** at which the current block was created.
- The **extraData** is an array of 32 bytes or fewer containing any surplus data pertaining to the current block.
- The **mixHash** used in unison with the nonce value proves that a viable amount of computation has been applied to the current block.

- The **nonce** value as previously stated is a random value combined with the mixHash to allow for a variation in the new hash value of the current block upon creation. In Ethereum's case a 64-bit value.

*This list items referenced from [25]

Along with these added block data objects came more ability to build data based applications on chain, in turn allowing for the addition of Smart Contracts into the Ethereum network.

2.3.2 Smart contracts

Smart Contracts are a fundamental part of how Ethereum operates being the combination of UI and protocols which allow for secure transactions on chain over a computer network and are a fundamental part of how Ethereum operates. Smart contracts are written in the object orientated language 'Solidity' proposed by Gavin Wood in 2014 and taken over by the Ethereum project for development as their language of choice. A smart contract in essence functions much like a real legal contract in the sense that until certain pre-agreed conditions set by the smart contract creator are met an the outcome of a transaction by parties involved will not commence or finalise. Smart contracts are the method in which developers interact with the Ethereum Virtual Machine(EVM).

Due to all parts of the transaction happening on chain , the transaction does not require a third party or overseer to maintain or guarantee its integrity. This is the trustless factor being demonstrated within the Ethereum architecture as all smart contracts are fully automated meaning no middle man fees, or liquidity providers are necessary. As all factors of the transactions or transfer or funds are agreed upon prior to the transaction starting the parties involved do not need to worry about the state of their funds as if all parameters of the transaction are met, it completes , otherwise all funds will return to their respective parties.

2.3.3 The Ethereum Virtual Machine(EVM)

The EVM is a quasi-Turing-complete state machine [25], this being the case due to the limitation on computations possible by the 'gas' parameter. *"In the field of computer science, a state machine refers to a machine that is able to read a series of inputs and, based on those inputs, transition to a new state"*[16]. The current state of the Ethereum virtual machine revolves upon whether transactions are being made as without new input no state transition will occur. For these transactions to effect the state of the EVM they

must be first verified by the Proof-of-work model or later , as explained in section 2.3.7, by the new Proof-of-stake model.

'Gas' is a unit of work done by the EVM and is a parameter that must be stated before a transaction can take place. Much like gas in a car the idea is that the computation ability of the EVM is limited to the amount of 'Gas' you have in the tank, or in this case that you have paid for prior to compilation and migration of your contracts onto the chain. With this idea came the concern that any transaction that runs out of gas mid way would be open to malicious tampering, however the EVM has been developed in such a way that in this exact case, if insufficient gas is provided the state of the block will return to prior to what it was the transactions start.

The EVM functions like a CPU as all contracts and message calls to the EVM must be sent as byte code from a compiler. In this case solidity is used as the compiler. The stack depth of the EVM is 1024, with each value stored on stack being a 256-bit word. This design was chosen as its storage method is aligned with the cryptographic hashing function used in Ethereum , keccak-256 [7].

The solidity compiled contracts , after being converted to bytecode , execute as a set of EVM opcodes. On the stack, operations such as AND, ADD, SUB are all still common place but EVM specific opcodes such as ADDRESS, BALANCE, KECCAK256, BLOCKHASH also exist and are used in contract calls to the EVM for transactions and timestamp calls [7].

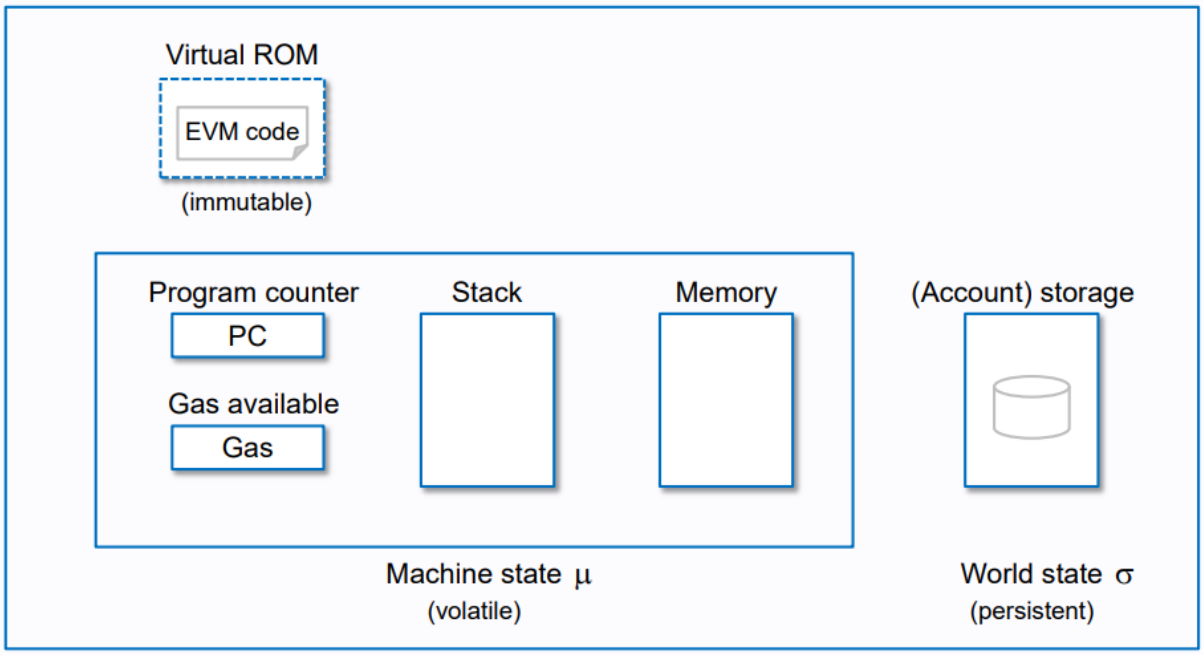

Figure 2.4: The EVM is a simple stack-based architecture

2.3.4 Gas Fee's and transaction costs

As Ethereum has exploded in popularity, currently running over three thousand Decentralised Applications on chain currently, the gas fee and transaction cost has also exploded since Ethereum's beginning. The unit of measure for gas in Ethereum is 'Wei'. One Wei is equivalent to 10^{18} Ether with the average gas price being measured currently in Gwei. One Gwei is equivalent to 1 billion Wei. The average gas price has fluctuated massively, moving between 30 Gwei and 211 Gwei in the last year alone, although paid in Ether, in FIAT currency being anywhere between 0.00006255 USD and 0.0004378 USD in Gas.

2.3.5 The Scalability problem

One of Ethereum's main issues right now lies within its inability to handle transactions at a speed matching that of the current demand. With an average transaction speed on ETH 1.0 of 15 to 45 transactions per second, Ethereum's throughput does not scale to the demand for transactions on the chain causing massive price hikes in gas fees and transaction costs. In this current climate, users trying to get their transactions mined and verified on chain bid with their gas fees for a place in the queue of miners. Miners then hash the transaction offering the highest amount of gas as it proves to be the most rewarding in terms of profit for the miner. As Ethereum's popularity rises and dips through the market cycle the already strained transaction speed of the network is pushed even harder causing extortionate gas fees and transaction costs for the average user no matter the amount of Ether to be transferred.

Currently the Ethereum development team are working on ETH 2.0, which hopes to massively alleviate the throughput issue that Ethereum faces currently on Layer One. With a proposed one hundred thousand transactions per second, ETH 2.0 hopes to scale the blockchain network to meet the current demand on chain. Moving from the current proof-of-work model to the new proof-of-stake validation model mitigates the "miner-monopoly" that currently exists where miners are the driving force of validation therefore taking the path of most profit rather then the path of most efficiency.

Proof-of-stake in comparison to proof-of-work is still in its early stages and has not been as war tested[reference the Eth web-page on pos] as proof-of-work. In Proof-of-Stake users stake their own Ether in one of two ways , either a user stakes a minimum of 32 Ether to become a full node owner or a group of users stake their Ether collectively in what is known as a staking pool, which cumulatively holds enough Ether to own a node on the network. The miners are no longer required as nodes act as validators for transactions and blocks on the network. Along with this no Ether is rewarded due to mining but instead node owners collect a percentage of the transaction cost of every transaction that

is validated by their node.

2.3.6 The Data Availability Problem

Throughout Ethereum's timeline an issue arose around the most successful Dapp deployed in 2017 "Cryptokitties". As the Dapp skyrocketed in popularity it highlighted an issue not yet addressed within the Ethereum Network. As the popularity of this Dapp grew so did the data requirements needed to sustain it. This in turn required the block data size on the Ethereum mainchain to grow at an alarming rate as the majority of the networks resources were monopolised by the Dapp. As the Layer Two solutions described in **??** were only in development, the chain data size has grown to almost one terabyte in present day shown in figure 2.5 illustrating the need to off load data from the mainchain. The issue arising with the chain size increase is a future in which the computational requirements, in terms of hardware required, for running a node on the network becomes too great. In this outcome centralisation could occur as entities monopolise the hardware required to participate as a node on the network thus nullifying the reason for Blockchain's creation, Decentralisation and no censorship. The question should arise as to what took the development of these Layer Two solutions so long, the answer being the data availability problem.

The data availability problem can be explained through the poker player anecdote. A poker player enters a casino and exchanges his money for chips. He enters at a table and plays the game with average success. The player wins a large hand and decides to cash out from the game but before he can he is knocked unconscious by the dealer and looses all memory of the last hand and his winnings. Before waking up the dealer and players have returned the chips you had won to their previous positions, pretend that you did not win the previous hand and continue the game as it was prior. You having to take their word on the matter move on with the game no longer having the chips you won.

In the example the cash for chips exchange is an on-chain transaction. The games played at the poker table are off-chain transactions. The player being unconscious and losing his memory of the hand is data now unavailable to him. As these transactions and malicious behaviours happen off-chain the transaction history you will return with to the mainchain will be valid and yet theft has occurred. This is the data availability problem that Layer Two solutions must address before any Layer Two solution can be regarded as secure hence the large wait time on any one solutions being deemed the most efficient without thorough testing [20].

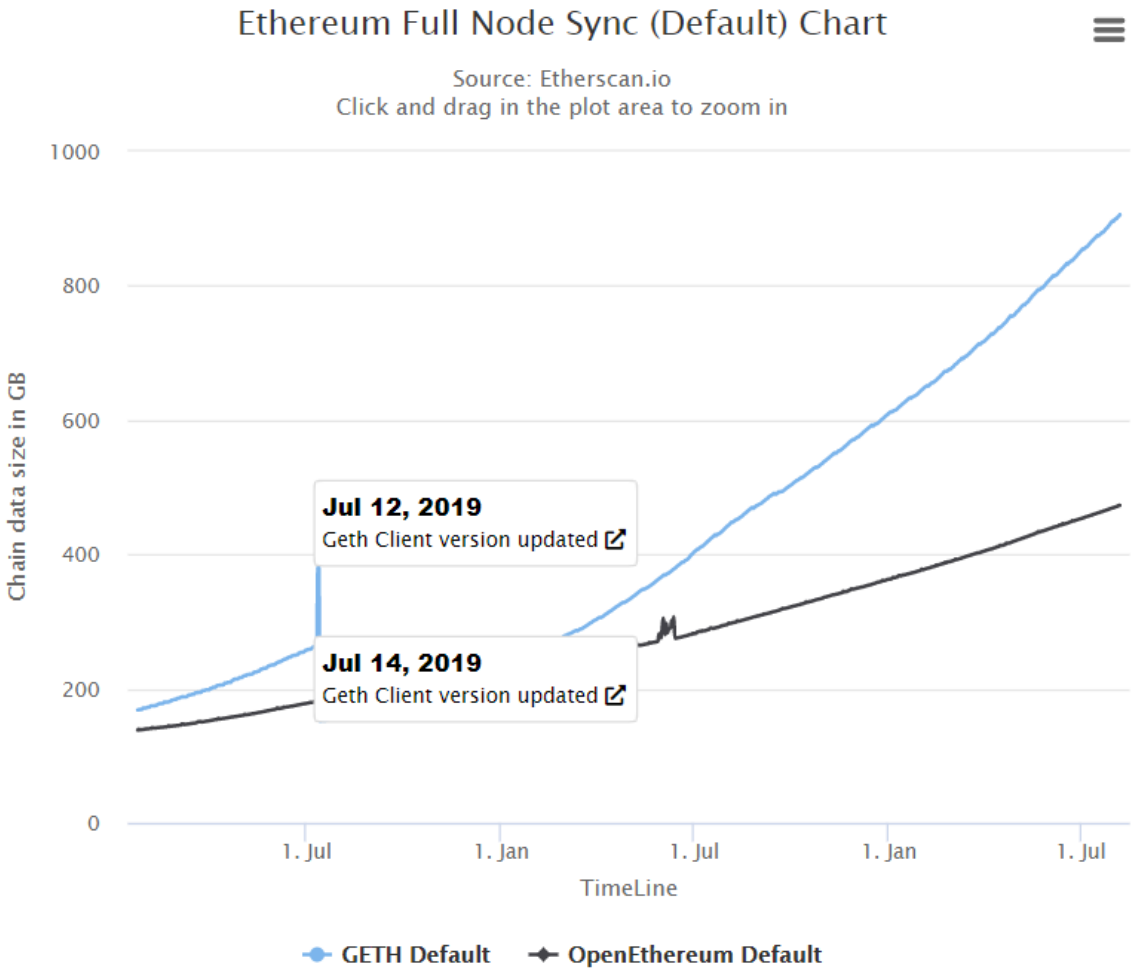

Figure 2.5: Chart showing the increasing demand and growth of data required by the Geth Nodes to hold the Ethereum Blockchain data

2.3.7 Ethereum 2.0

The Current development happening within Ethereum as stated previously is the move from Proof-of-work to the Proof-of-stake protocol. This comes with great benefits, but has also raised many community concerns about the future of the Ethereum network. With Ethereum 1.0 being bottlenecked by its 15 to 45 Transactions-per-second(TPS) it is in need of an infrastructural change to begin to meet demand and lower the exuberant gas fees surrounding its transaction cost issue. Understanding the changes to come along with Ethereum 2.0 is important due to the nature change of the Layer Two solutions that will still be present after the move has been finalised. As it stands at present the changes coming with Ethereum 2.0 are as followed:

- A preexisting concept within other blockchains partitions the set of nodes into multiple smaller groups of nodes called committees that operate in parallel on disjoint blocks of transactions and maintain disjoint ledgers. Such a partitioning of operations and/or data among multiple groups of nodes is often referred to as *sharding* [26]. Built upon this idea is *Shard Chains* for ETH 2.0. Shard chains spread the network's load across 64 new chains. They make it easier to run a node by keeping hardware requirements low [9]. As the requirements for running a node become less this garners a healthier network as the ability to run a node becomes more accessible to more users. With sharding the dramatic rise in memory required to store the current chain state should ease substantially and lower the rate of growth to a fraction of what it is currently.

- As Ethereum is upgrading to 2.0 the Proof-of-work model will be changed to a Proof-of-stake model thus significantly decreasing the amount of energy required to run a node on the chain.

- The move to ETH 2.0 cannot be instantaneous therefore a gradual merge between 1.0 and 2.0 must happen. The Beacon chain was launched on December first 2020 with its purpose being to begin the move towards Proof-of-stake, allowing staking of Eth and use of the new security features that come with it.

- The move from proof-of-work to proof-of-stake comes inherent security changes within the Ethereum network. The transition to proof-of-stake means that the Ethereum protocol has greater disincentives against attack. This is because in proof-of-stake, the validators who secure the network must stake significant amounts of ETH into the protocol. If they try and attack the network, the protocol can automatically destroy their ETH proof-of-stake [8].

2.4 Layer Two Solutions

Layer Two solutions all center around the general idea of being a side-supporting application to the main chain. This can be an alternate blockchain entirely or an internal protocol within the original blockchain. This method of "off-loading" work from the main chain to a side chain or protocol has shown promise in the past as we have a current working example on Bitcoin known as the "Lightning Network". As an already standing example of a Layer Two solution already implemented and working efficiently, it gives merit to the idea that this area is where Ethereum will develop further alongside the release of Ethereum 2.0. While researching for this project many of the current available options for a Layer Two solution for the Ethereum Network were explored.

2.4.1 Channels

Channels allow for multiple transactions off-chain while only two transactions happen on the main chain of the network. In the case of Ethereum, the load is taken off of the EVM for these transactions as the only transactions that are handled on-chain are the sending and finalisation of the transaction. This greatly reduces the cost of computation on the main chain and allows for a higher level of throughput. Channels however comes with drawbacks as certain prefixed requirements are required for it to function correctly.

- Participants need to be known prior to the first transaction on main chain.
- Participants are required to deposit funds for the transactions into a "Multisig" contract. Multisig contracts or wallets work off the basis of multiple parties having to agree and digital sign a transaction before any transaction can take place. This greatly reduces any single-point of failure but does add additional steps to the process of making any transaction [2].
- Time is required prior to transactions to setup channels for each user decreasing ease of use for the end user and reducing open participation. Open participation referring to the ability for anyone at any stage before the transactions process begins to join the Multisig contract without prior arrangement.
- While the Multisig contract is being digitally signed prior to the transaction being instantiated on the network , the funds sit inside of the contract which in turn requires that the funds and contract have to be monitored constantly by the network before the contract is completed and validated on chain adding to the resources consumed.

2.4.2 Plasma

"A plasma chain is a separate blockchain that is anchored to the main Ethereum chain, and uses fraud proofs (like Optimistic Rollups) to arbitrate disputes. These chains are sometimes referred to as "child" chains as they are essentially smaller copies of the Ethereum Mainnet. Merkel trees enable creation of a limitless stack of these chains that can work to offload bandwidth from the parent chains (including Mainnet). These derive their security through fraud proofs, and each child chain has its own mechanism for block validation" [5].

Plasma being the parent of Optimism holds many similarities in terms of functionality. These similarities were necessary to research in depth to understand the base layer upon which Optimism was built. It's use of fraud-proofs opens windows of allotted time after a transaction has been finalised, but prior to it being verified on the main-chain, where any node on the Layer One network can challenge the validity of the transaction hash. If the transaction is proven by an internal node on the network to be invalid it becomes costly for the original submitter to re-compute a valid hash for the transaction and in this way discourages malpractice on chain. However if no node on the network challenges or can prove the hash is incorrect within the proof window the transactions are taken as valid and submitted to the main chain.

Plasma uses a Merkle Tree model as the basis for its scaling. As shown in figure 2.6 each Plasma block contains a transaction with its own hash. In an ideal situation where an even number of transactions are to be computed the hash of block A and block B will be added creating an AB hash. In the same way each non computed block will be paired with its succeeding block ie. CD, EF, GH, etc. to give a combined block hash. This methods continues upwards, forming ABCD block hashes and so on. When all transactions have been entered into the tree the finalised single hash at the top of the tree is taken as the root hash or the "Merkle Root" and this is submitted to the main chain for verification.

The merkle tree method expands onto a merkle proof. In the case of withdrawals, when a user interacts with the root chain contract

In general Plasma is encapsulated by the following:

- Plasma maintains a high throughput as well as a low Gas price per transaction validated.
- Functionally efficient for transactions between users already established on the Plasma network.
- Although it maintains an efficient peer to peer transaction cost, the network does not support general computation ie. only token transfers and swaps are supported on the network.
- Requires the use of Operators on the network to store data and serve it when requested.
- Withdrawing funds from the network, much like Optimism, requires a withdrawal window. This time span is usually several days but can be scaled upwards based upon the current demand on the main network. In this way, although it delays the withdrawal of funds quite severely, it mitigates the issue of a mass withdrawal of funds in a short period of time known as a "Plasma Exit" causing a huge bottleneck on the main network. This in turn would exponentially rise gas prices for users on mainnet defeating the purpose of the Layer Two Solution.
- Although its efficiency due to the use of child chains created from the Merkle tree method allows for near infinite scaling, the computation limitation of the Plasma network does not allow for general purpose smart contracts found on the Layer One Ethereum mainnet, making it an imperfect solution for all use cases.

*All list items information gathered from [17].

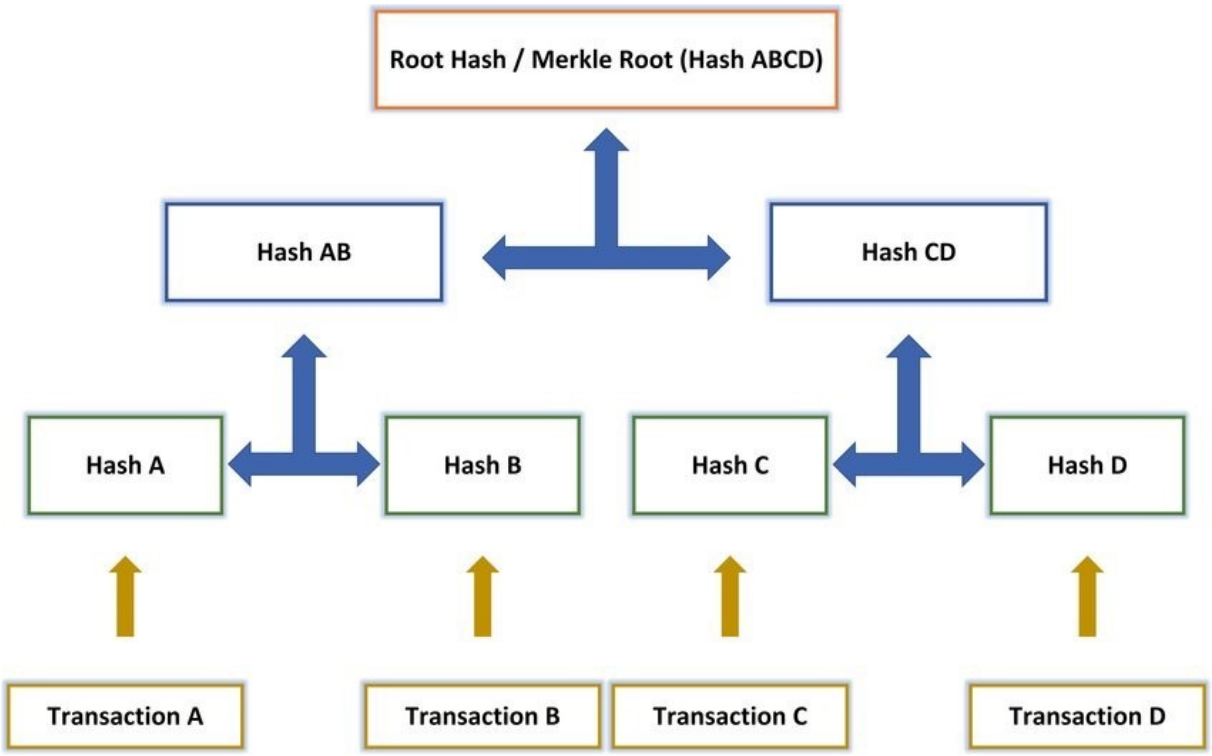

Figure 2.6: Example diagram of the Merle tree method used in Plasma

2.4.3 Sidechains

Sidechains are completely separate and independent from the main chain. They contain their own consensus models and hash proof protocols to validate Layer One transactions. Although they have different protocols to the main chain they are main chain compatible in that, in Ethereum's case, all sidechains must use the EVM as their computational machine allowing any smart contract that could be deployed to the main Ethereum network to also be deployed with full functionality to said sidechain. Sidechains are given more freedom in terms which security and proof methods are to be used by their chains but they also run into the issue of being less decentralised than the Ethereum mainnet. Theoretically on a sidechain, validator nodes on the network could potentially group together to perform a fifty-one percent attack.

"The fifty one percent attack is a technique that occurs when an attacker is in possession of fifty one percent of the hashing power. This attack starts by creating a chain of blocks privately, which is fully isolated from the real version of the chain. At a later stage, the isolated chain is presented to the network to be established as a genuine chain. This is what enables the double-spending attack. Since the blockchain policy complies with the longest chain rule, if attackers are able to get fifty one percent of the hashing power or more, they will be in a position to drive the longest chain by persuading the network nodes to follow their chain. However, it is not strictly necessary to obtain fifty one percent of the hashing power; if attackers get less than half of the hashing power, the double-spending attack is still possible but with less probability of success. The more hashing power the entire blockchain network comprises, the more costly the attack becomes. Thus, cryptocurrencies with high network hashing are assumed to be more secure against the fifty one percent attack" ([21]).

This issue is not exclusive to a sidechain but becomes more feasible and monetarily appealing to node owners on sidechains that carry large monetary worth on the network. An example of this is the Bitcoin Gold fifty-one percent attack of 2018 and 2020 where eighteen million dollars worth of Bitcoin gold was stolen due to the roll back and double spend abilities made possible by the attack.

2.4.4 Rollups

Rollups are a general purpose scaling solution that maintains its security level by keeping in-line with the Layer One consensus and security protocol model of the mainchain, in this case Ethereum. Rollups have become the more popular choice in terms of scaling Ethereum, now and after Ethereum 2.0, for these reasons. The basic idea behind a rollup is not very different from its other Layer Two scaling brethren, off-loading work from the mainnet to a sidechain that does all transaction computing and passes the finalised transaction hash back to Layer One chain for verification. How rollups differ from the previously mentioned options is in their ability to maintain most, if not all of the functionality of the Ethereum mainchain as well as keeping the Layer One security model intact.

generally rollups can be summed up through the following points:

- Maintain the same consensus model as Ethereum.
- Maintain smart contract functionality on the layer 2 network with direct compatibility with the EVM.
- Rollups group together transactions of a user while on the layer 2 network, hash all the transactions on Layer Two without returning to Layer One, and push the finalized hash value from the bundled transactions to Layer One for verification on the chain.

2.4.5 Optimistic Rollups VS Zero Knowledge Rollups

As part of the project, research was put into all rollup options available at the time of starting, these being Optimistic Rollups and Zero Knowledge Rollups(ZK-rollups). Although functionally similar, huge differences in the inner workings of these two options and the approaches to solving the scalability issue made for interesting research and affected the choice of rollup technology implemented later in the project.

The decision between using ZK-rollups and an Optimistic rollup came down to multiple factors, the first of which being the development stage that ZK-rollups were in at the start of this project. Previous to the now fully functioning EVM support, ZK-Rollups could not handle the majority of EVM supported smart contracts meaning that direct translation from Ethereum mainnet to a ZK-rollup was not a fluid task. In comparison, the Optimistic Virtual Machine(OVM) showed almost perfect interoperability between the EVM and OVM allowing for fluid smart contract deployment with minimal to no solidity code changes required. Optimism has currently launched both a Kovan testnet

and an Optimism mainnet which was the original proposed location for testing in this project whereas ZK-rollups were a much less streamlined testing experience at the time of selection.

2.5 Zero Knowledge Rollups

ZK-rollups are the combination of the rollups concept along with previously used and demonstrated working models of a security protocol known as ZK-SNARKS. Projects such as ZCash have already implemented this working model. *"Zero knowledge rollups, also known as ZK-rollups, bundle or "roll up" hundreds of transfers off-chain and generates a cryptographic proof, known as a SNARK (succinct non-interactive argument of knowledge). This is known as a validity proof and is posted on Layer One"* ([4]). Much like a zero knowledge proof described in section 3.3, a SNARK allows for a 'prover' to prove that they know the secret required for the transaction without giving away that secret publicly, the difference being that within a SNARK no interaction between the Prover and Verifier is required.

2.5.1 Proof Method

ZK-Rollups employ the "Zero Knowledge Proof"(ZKP) method. In this method two parties, the 'prover' and the 'verifier', exist. When deployed correctly a ZKP can be used to verify that the information within a transaction is true and valid without either party having to disclose their private information. In terms of explanation the "Alibabas secret cave" example proves easiest for understanding. In figure 2.7 we see a female and male, Amy and Bob at the mouth of Alibabas cave. In the cave there are two pathways, A and B. At the end of both pathways there is a door that requires a secret to open the door. Amy, the prover, wants to prove to Bob, the verifier, that she knows the secret for opening the door without publicly releasing that information. Bob wants to verify the trust value of Amy by verifying whether she does know the secret. Amy enters the cave while Bob waits outside. Amy chooses a path randomly and arrives at the door. Bob now enters the cave and informs Amy as to what pathway back to him he wants her to arrive from. Amy , knowing the secret, then opens the door and returns along the path chosen by Bob. However the possibility that Bob chose the path Amy had chosen to travel initially and that she simply returned back the same path exists so the test must be ran multiple times. The probability of Amy being dishonest and guessing the pathway Bob would choose in a ZKP, ran twenty times, is approximately one in a million. Therefore we can assume that Amy is honest and does know the secret required to open the door

without revealing the secret.

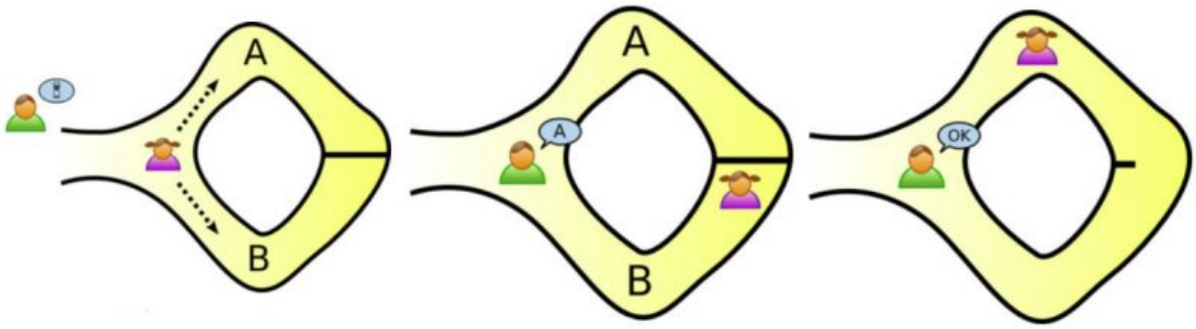

Figure 2.7: An example figure of Alibaba's cave, Amy being the female figure and bob being the male figure, both pathways A and B aswell as well as the door at the pathway intersection.

Built upon this, we revisit SNARKs as the functioning working model for ZK-rollups. A Zero-knowledge succinct non-interactive argument of knowledge(ZK-SNARK) is the combination of the previously described protocols. The mathematical equation for a ZK-SNARK can be described as

$$\boldsymbol{Y = C(x,\ w)} \tag{2.1}$$

where $\boldsymbol{Y}$ is the public output. $\boldsymbol{C}$ is the public function. $\boldsymbol{x}$ the public input and $\boldsymbol{w}$ the secret input.

2.5.2 Scalability

ZK-rollups are computationally quite taxing, in terms of hardware a test ran in 2018 on a laptop with 7GB of ram available and a 20GB swap space would struggle to average 20 TPS in comparison to ETH 1.0's 15-45. In more recent times, namely the last year, however ZK-Rollups have been optimized massively now with a theoretical 2140 TPS [11]. The SNARK protocol used is based upon the Groth protocol [12]. the Groth protocol introduced pairing-based non-interactive zero-knowledge proofs, yielding the first linear size proof based on standard assumptions [13]. In this way it is possible to have a verification complexity of $\mathbf{O(1)}$ which in theory means consistent costs regardless of the amount of transactions processed [12]. This is possibly further optimised through the change from Central processing units(CPU) to a Graphical-processing unit(GPU) as the GPU, with its thousands of cuda cores, seem better suited to processing this type of algorithm.

The initial limitation on ZK-rollups were its requirements for a trusted setup. Initially the system would need to be quite centralised to work efficiently. A move to proof-of-stake is required to make full use of the ZK-rollups as it is required for preventing a single

operator on the network with more efficient proving hardware than others from censoring data on chain as well as punishing operators that do not deliver promises in an attempted Denial of Service(DOS) attack by burning their staked Ether as collateral.

Currently ZkSync, produced by matter labs, is a live ZK-rollup on the Ethereum mainnet with multiple full feature sets available. The system does use a new framework called Zinc, which in language terms closely resembles rust. Although the framework gives ability to the ZK-rollup currently it is not a mass adopted language and smart contract support for preexisting solidity contracts is not completely interoperable between the EVM and a ZK-Rollup at this time.

2.5.3 Security

The overarching debate in the space of rollups right now is between fraud proofs and validity proofs. Where fraud proofs require a "timeout window" for a challenger to challenge the hash output from layer 2 , the Validity proof requires the computation and completion of a cryptographic proof to assure validity. Previously fraud proofs were looked upon as the only viable path forward as validity proofs were computationally extremely expensive making them the less appealing rollup security method. However with the latest developments in the space the computation has been optimised greatly as well as the idea of using GPUs for further optimisation of the computation. In terms of security a ZK-rollup has a more secure model basing its security on its cryptographic proofs however this impacts the scalability of the system as a proof is required for each state transition which is computationally taxing and limits throughput.

Along with all forms of current blockchain computing there is always the future issue of a quantum computer which can calculate hash and nonce values quicker then the nodes or miners on a network which would give rise to the aforementioned 51 percent attack. As well as this a quantum computer with this capability would allow for double spending, invalidating valid blocks and invalid transactions to possibly occur.

As is the case with all rollups they have an inherent decentralisation as transactions are finalised on Layer One , giving them the same level of decentralisation as is available on the mainchain. However as they require smart contracts on Layer One to act as both a wallet for locked funds and a gateway into the layer 2 smart contract, this structure in itself creates a"centralisation" of funds. The gateway smart contracts running the Layer Two solution become somewhat of a "honey pot" posing a potential security issue in the future.

Chapter 3

Optimism

The primary purpose of this project is a high level exploration of the Optimism space. Within this section the topics of how the system functions, the Optimistic Virtual Machine, the Proof methods employed within the network, the Scalability that optimism can provide as well as an in depth look at its Security protocols are dismantled and explained. Optimism is the network on which Optimistic rollups have been deployed and is also the development network for Optimistic DApps. The main appeal currently with Optimism is its theoretical ease of use for porting already existing DApps from the Ethereum mainnet to its layer 2 network, although in practice throughout the project it was found to be lacking in the area somewhat. Although its TPS is somewhat slower than that of its counterpart ZK-Rollups, this ease of use porting of general purpose smart contracts is the driving factor as to why Optimism garnered such support from the community.

3.1 The System

Optimism, although called a layer two, functions inside of the Ethereum Blockchain. This is done through the use of Smart contracts that emulate a modified version of the EVM. This emulated and slightly tweaked version of the Ethereum Virtual Machine is known as the *Optimistic Virtual Machine* which will be explored in section 3.2. For a user moving from the mainnet to the optimistic network, the steps are as follows:

1. A user will decide upon an amount of Ether to be transferred to the layer 2 optimistic network.

2. This amount will be exchanged using the Optimism Gateway found at : *https://gateway.optimism.io/*

3. The amount of funds chosen will be deducted from the users Ether wallet and

locked into the Optimistic Layer 1 Gateway smart contract. This contract functions somewhat as a wallet for all funds currently tied up in layer 2. This contract also works somewhat as a liquidator for the wallet itself as a place to pay out withdrawals after a user returns from layer 1 to layer 2.

4. A second contract works as an emulated EVM , known as the OVM. In the process known as *minting*, an equal number of Optimistic Ethereum to what was deposited from layer 1 are created. For example if Alice deposits one-hundred Ether into the Optimistic contract then, after locking the initial 100 Ether onto layer 1, one-hundred optimistic Ether tokens will be created and allotted to Alice for spending on layer 2.

5. The user makes transactions on the network in the forms of contract deployment, contract calls or normal peer-to-peer transactions using the Optimistic Ethereum tokens. These transactions are bundled into a optimistic rollup along with a number of other transactions occurring at any given time.

6. this rollup of transactions is then computed, hashed and that hash output is returned to layer 1 via the smart contracts. This output hash is then input into the Fraud proof system explained in section 3.3 and shown publicly for a Dispute Time Delay(DTD). Within this proof window, of usually seven days, any verifier node on the network may challenge the output of this rollup. If no challenge to the output of the rollup proves true , the output is accepted as valid and posted on layer 1. If a challenge proves true, then the node that put forth the incorrect has value is punished by burning some of the staked Ether required to own a node explained in 2.3.7.

7. When a user has finished on the layer 2 network and wishes to withdraw to layer 1, the current balance of Optimistic Ether in the users possession is burned by the system and the same amount of Ethereum is returned to the user from the Smart contract on layer 1 mainnet. This withdrawal is also subject to the fraud proof window meaning it takes seven days to withdraw funds back to layer 1 in the optimistic security model.

Multiple Smart contracts form the basis on how Optimism functions. A simplified diagram for the sake of explanation is shown in figure 3.1. A high level diagram of the OPtimistic Ethereum Structure can be found in the Abstract(6.2) in figure 1. This example is of a simple use case flow of the layer 1 to layer 2 transition and layer 2 returning to layer 1. All black text are functions moving from layer 1 to 2, red being layer 2 to layer 1.

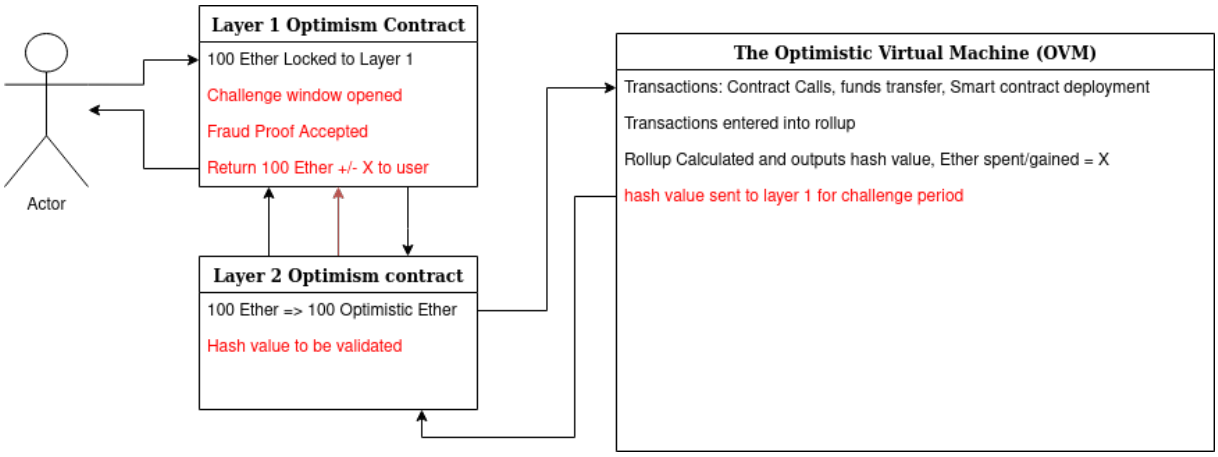

Figure 3.1: Example diagram of a simplified user interaction with the optimism network and rollups

3.2 The Optimistic Virtual Machine

State transitions are a state change that happens within the EVM every time a transaction occurs. As Optimism aimed to prevent invalid state changes from occurring off chain,as this would result in invalid transactions when returning to Layer 1, they created the Optimistic Virtual machine to function within the EVM so as to inherit its security protocols and structure ensuring layer 1 safety on layer 2. This however is a difficult task as differences between values such a the block timestamp on layer 2 create errors in output which breaks the system. Within this section the modifications and changes required by the OVM to mitigate and solve these potential errors are explored.

3.2.1 The Optimistic future Cone

Ethereum states are the set of all Ethereum state transitions possible from the current state of the Blockchain. At each state the EVM calculates the next possible moves for the system and places them inside a "cone". A clearer example of this can be seen in figure 3.2 and 3.3 as from the diagram the finalised blocks in green have already been transitioned through by the EVM whereas the yellow area highlighted within the EVM shows the set of all possible state transitions from this point forward. The red space illustrates the state transitions no longer possible and therfore discarded by the system. This cone concept is important as it is modified within the OVM to improve speed and compute costs of these transitions.

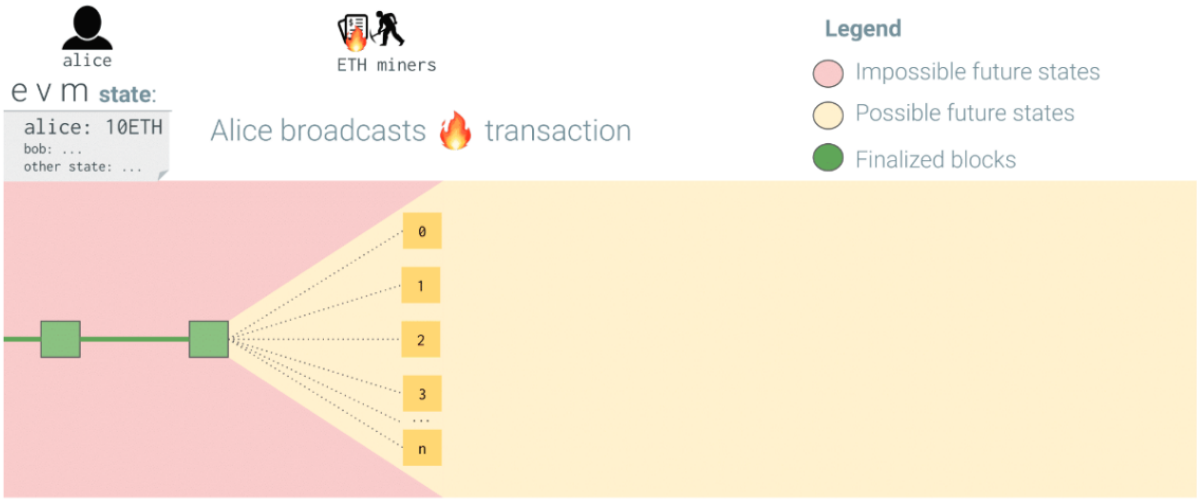

Figure 3.2: Example diagram of the Ethereum future cone determining the possible state transitions for the Blockchain

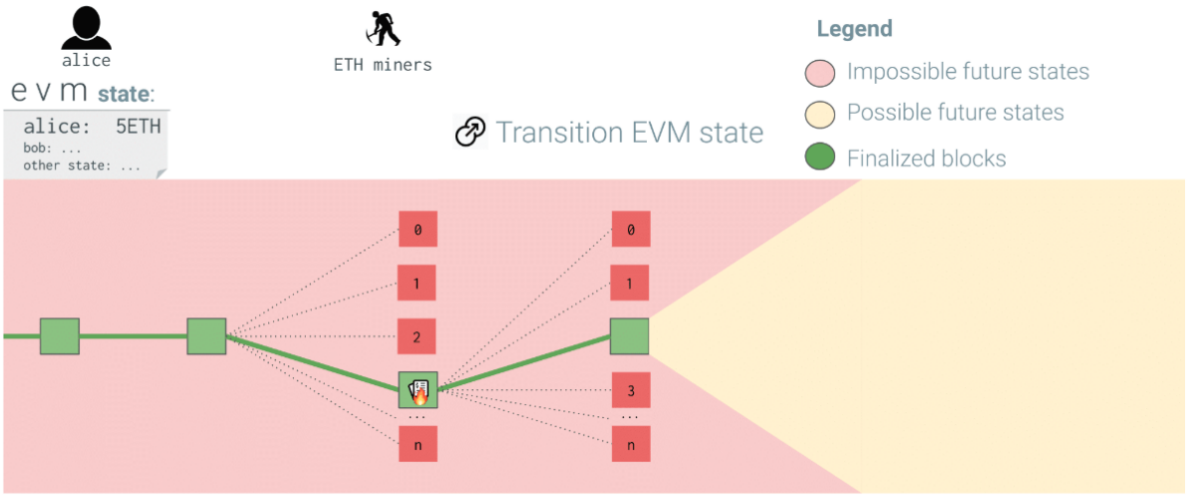

Figure 3.3: Example diagram of the Ethereum future cone determining the new possible state transitions for the Blockchain

The optimistic network builds upon the Layer 1 consensus protocol by gathering its own localised information around the current state. *"Layer 1 (L1) gives us a trusted, but expensive, virtual machine (VM). Layer 2 provides an interface for efficiently using the expensive L1 VM — instead of transactions directly updating the L1 state, we use off-chain data to guarantee what will happen to L1 state. We call this guarantee an "optimistic decision.""*[19]. This gives rise to the topic of local assumptions made off chain to guarantee potential state transitions. An example of this in action is the "displute liveness assumption". As the OVM gathers information about the transactions occurring, in the form of off-chain messages [14] it defines the assumptions required to make an "optimistic" decision on which states are possible given the information. In this way since the OVM can assume that in any given channel, since the fraud proof windows exist, any state in which a malicious withdrawal went undisputed or detected is a non viable possible state transition and therefore can be removed from the set of all possible state transitions. This narrows the "cone" of possible transitions allowing for less computation to be required. This can be seen in fig 3.4 where through assumptions made on local information , states 0 and 4 - $\boldsymbol{n}$ can be optimistically removed from the potential state transitions. When the OVM's state transitions forward from this point, as more data becomes available, it further limits the states possible therefore decreasing the level of computation required by the OVM.

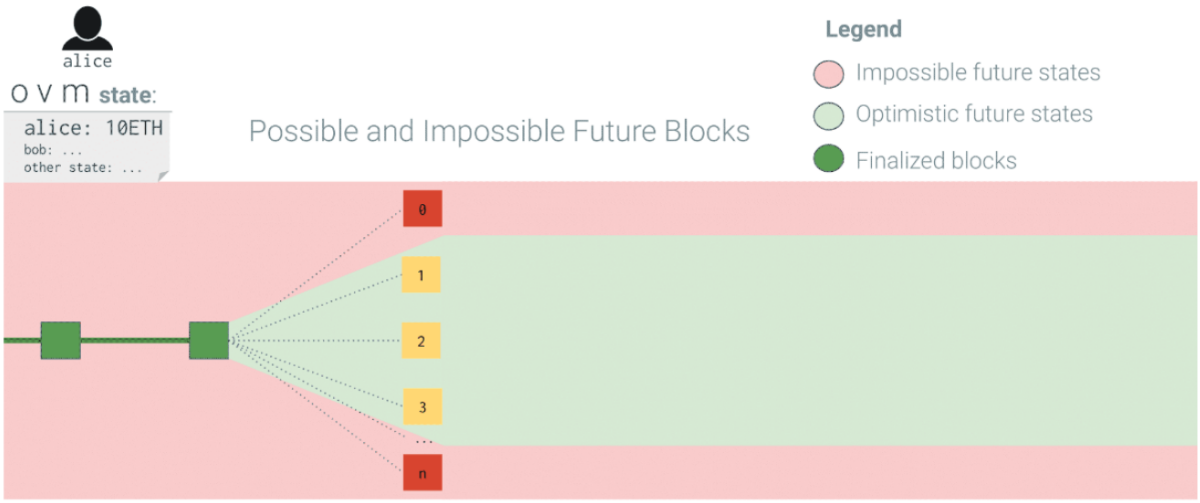

Figure 3.4: Example diagram of the Optimistic future cone determining the possible state transitions for the Blockchain

The Optimistic Virtual Machine, as explained previously is a modified EVM running on the optimistic network. The OVM allows for interoperability between Layer 1 EVM general purpose smart contracts and the Optimistic network. The OVM is compatible with all EVM functions and solidity compiled smart contracts. The OVM exists to allow

for smart contract execution between layer 1 and layer 2. *"The OVM guarantees deterministic smart contract execution between L1 and L2"* [15] by being able to reprocess any transaction on layer 2 back on layer 1 to prove its validity. Any input given to the OVM in the form of opcode will only have one outcome, the computational trace of the OVM transaction will be identical to that of an identical transaction re-calculated on layer 1. This is crucial in design as the Fraud proof model and its proof window function through the ability for any layer one node to recalculate the transactions from the layer 2 rollup, having access to the transactions and any other pertinent information by use of the merkle tree system, at any time back on the layer 1 Ethereum mainnet. This ability to guarantee correct data availability between layer 1 and layer 2 comes from the "Execution manager" within the OVM.

3.2.2 The Execution Manager

Within the OVM lies the answer to the data availability issue between layer 1 and layer 2. Once again a smart contract gives greater functionality to the system in the form of the Execution Manager. The Execution manager is a virtualised container for OVM smart contracts [6] and virtualises any and every data point that may cause differences in execution of a transaction on Layer 2 from its EVM counterpart on Layer 1. This includes:

- Smart contract storage.
- The "Context" attached to any layer 2 transaction ie. block number, timestamp, the address of the account which sent the transactions (tx.origin).
- Routing of messages between smart contracts within the OVM.

In essence the Execution Manager virtualises any function of the EVM that may produce a different result on Layer 2. There are approximately 15 Ethereum instructions virtualised inside the Execution manager currently [6], one of which being the *block.timestamp* Ethereum function. This simply returns the timestamp of a block when verifying its transactions on the EVM. This however will not function on the OVM as a difference in timestamp between the two chains will incur different outputs when verified back on layer 1 during a fraud proof as a *block.timestamp* function call will return the timestamp of the current block , not the block in which the transaction took place on layer 2. The Execution manager mitigates this issue by constructing a container which correctly feeds the appropriate block timestamp in the form of *timestampManager.getOvmTimestamp()* back to layer 1 so that the computations are identical on both chains.

3.2.3 The Purity checker

An added function of the OVM is its "Purity checker". As explained in section 3.2.2 the *block.timestamp* issue is mitigated through the execution manager container, however as general purpose smart contracts are deployed to the OVM they must be guaranteed to use the OVM's virtualised EVM functions or else they become security risks. The Purity checker inspects whether a smart contract attempting to deploy to the OVM has only made the correct virtualised function calls thrgough the Execution manager and not EVM specific function calls such as *block.timestamp*. If no EVM specific functions are used then deployment is successful and the Smart Contract is deployed to the optimistic network, if EVM specific functions are found within the solidity code of the smart contract then it is rejected guaranteeing Layer 2 safety.

3.2.4 The Transpiler

As the Execution manager and purity checker mitigate the security issues revolving around using the EVM specific functions mentioned previously, this does not gurantee an ease-of-use or fluidity between deploying from Layer 1 to the optimistic Layer 2. This is where the Transpiler comes into play. The transpiler takes in normal EVM bytecode and converts it to OVM bytecode using container functions calls. This means that developers do not need to scrape through their already existing EVM compatible smart contracts and replace each EVM function call with a OVM alternative to make this migration successful. The Transpiler can be added to the developers development testing suite and will automatically convert any EVM function calls to their OVM alternative.

3.3 Proof Method

As previously explored in the case of Plasma(2.4.2), the method of fraud proofs has been further expanded within Optimism and its Optimistic Rollups. In contrast to ZK-rollups validity proof, which proves a transaction is valid within its ZK-SNARK, the fraud proof is a an attempt at proving that a transaction is valid by contradiction. As transactions are bundled together on Layer 2 they are computed by the OVM and returned to layer 1 as a single hash value. Within the Optimistic network there is state "liveness", this means at any given time a "watcher" node on the network will be monitoring a rollup bundle for anything malicious or incorrect. The fraud proof method functions with this by opening a timed Dispute window so that any watcher node on layer 1 can view the finalised hash value from the rollup. This dispute window is set to 7 days for Optimism to achieve maximum security. Within the 7 days any layer 1 node can challenge the output

from the optimistic rollup by recomputing all transactions from the rollup and verifying that the original hash output from the rollup is either correct or incorrect. A fraud proof implies a proof by contradiction in the sense that if no watcher node on the network can prove that a Optimistic Rollup output hash is incorrect then we can "optimistically decide" that the hash output must be valid. the hash value is then posted back to the Ethereum blockchain. In the case of an incorrect has value found within the dispute window the node responsible for finding the incorrect hash is rewarded with Ether and the node responsible for posting an incorrect hash is punished by burning, which is the word used to describe the destruction of Ether assets, their staked Ether within the node.

3.4 Scalability and Performance

The performance of a layer 2 solution is measured in 2 sections:

- **Throughput:** The transactions per second of the layer 2 solution
- **Latency:** time taken for a transaction to be processed on the network

Optimistic rollups take the path of reducing the cost of already existing functions on Layer 1, in doing so they provide the same layer 1 functionality and security at a cheaper computational cost and a lower gas fee. With the creation of EIP-2028 and the *Istanbul hard fork* of the Ethereum mainnet came the decreased *"calldata"* costs, namely moving from 68 gas per byte of data used to 16 gas per byte [1]. With this four times decrease in gas required came improved scaling for layer 2 solutions like Optimism. As calldata is the minimum information required to recreate transactions on Layer 1 from layer 2, we see a large TPS increase from Ethereum's 15 TPS to approximately 400 TPS on Optimism. While this is a large improvement it does not come close to the average TPS of something like VISA which hits an average of 2000 TPS. With the idea behind all of Blockchain being to decentralise finance, this outcome wont be possible until blockchain can keep up. This leads to the discussion of Boneh-Lynn-Shacham(BLS) signature aggregation within an Optimistic Rollups. BLS signature aggregation can be explained simply as taking the multiple signatures inside of the rollup and condensing them into one large signature that verifies that all parties included in the transaction were present and signed the transaction [3]. Inside Optimistic rollups transaction data there exists five fields:

- **From:** The signatures of the addresses from which funds are sent during the transactions.(65 bytes)
- **To:** The address to which the funds are sent during the transactions.(3 bytes)

- **Value:** The Amount of Ether to be sent during the transaction.(4-6 bytes)
- **Fee:** The transaction fee imposed upon the transaction by the system (1 byte)
- **Nonce:** The Nonce value for hash randomisation.(2 bytes)

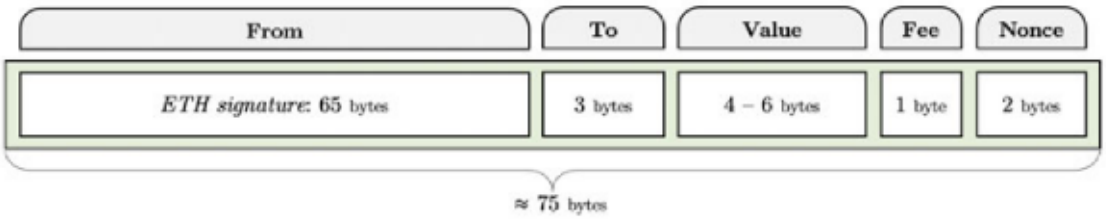

Figure 3.5: Transactions data layout within an Optimistic rollup

This Optimistic rollups transaction sums up to approximately 75 bytes of data required shown in figure 3.5. This increases the transaction size of Ethereum as the **From** field within a layer 1 transaction contains the single address from which funds are being sent. This increase in data size although large is balanced against the large computational benefits created from the use of *calldata* by the Optimistic Rollup. However it has been theorised that with BLS signature aggregation that this 65 byte large Elliptic Curve Digital Signature Algorithm(ECDSA) signature could be replaced by one BLS signature theoretically reducing the initial 75 byte transaction size back to its original approximate 13 bytes shown in figure 3.6. This in turn theoretically increases Optimisms throughput to approximately 2000 TPS [10]. This brings us into the realm of VISA's TPS although these are theoretical numbers and any bottleneck may present itself in the developments future. Along with the implementation of ETH 2.0 , Sharding explained in section 2.3.7 will give even greater data availability to the layer 2 solutions as well as greatly reducing the cost of data further. The performance increase that comes with this is yet to be seen.

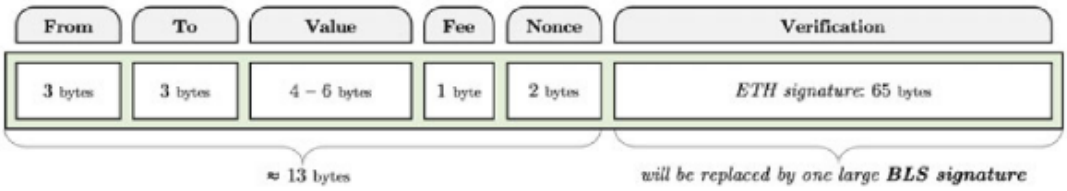

Figure 3.6: Transactions data layout within an Optimistic rollup after BLS signature aggregation

3.5 Security

As Optimism is layered within layer 1 it inherits Ethereum's consensus protocol, its cryptographic security as well as its security protocols and node structure. This gives great weight to the communities belief in Optimism as a layer 2 solution. However , there is currently a capital efficiency versus security debate ongoing. Optimism's fraud proofs and dispute windows, although verifiably secure, have a long withdrawal time attached to this security model, namely 7 days. As discussed, in this time any watcher node on the network can attempt to challenge any rollup transaction hash that they deem incorrect or unchecked. Here lies the heart of the issue, the wait time for withdrawal is too long. Many users will move to a ZK-rollup, which in comparison currently, will take thirty minutes to compute its ZK-SNARK and the transaction on layer 2 will be finalised allowing funds to be withdrawn back to layer1. This incentivises Optimism shortening its Dispute time Delay(DTD) of 7 days in order to compete in the layer 2 space but this opens Optimism up to Fraud and censorship attacks due to the cut in security and the large reduction in cost it would take for a malicious user to rent enough compute power to complete fraudulent transactions without being noticed on the network. This issue is possibly mitigated by the use of liquidity providers or exit markets on the layer 2 network. In the case of Alice wanting to withdraw 5 Ether from Layer 2 but not wanting to wait through the 7 days proof window she uses a liquidity provider on the layer 2 network. This liquidity provider or market, as they too are on the layer 2 network can use the localised information on the chain to prove that Alice's transactions are non fraudulent and that her withdrawal will prove valid on Layer 1 after being challenged. Therefore the liquidity provider offers Alice 4.9 Ether instantly to her own Layer 1 wallet withou needing the wait period of the proof window. Alice accepts and receives 4.9 Ether to her Layer 1 Ethereum wallet. The liquidity provider gains 0.1 Ether as commission for their services [24]. The fundamental downside to this solution is that it is not inherent to the system ie. it relies upon outsourced funds and development of such a marketplace. It also doesn't not solve the time frame issue but rather just improves the user experience around it to a varying degree. Another issue with this solution is that a liquidity provider's funds would need to be locked on the Layer 1 smart contract in order to facilitate these services which brings us the Honey Pot Issue.

3.5.1 The Honey Pot

As users move from Layer 1 to the Optimistic Layer 2 network , their original Ether funds deposited to the network are locked onto the Layer 1 smart contract. As more users come online and lock their funds to access the network, the amount of Ether stored

in one location on the network grows exponentially. This is a prime candidate for the term Honey pot as this inherent localisation of funds somewhat deviates from the idea of decentralisation as well as painting a bulls-eye on the back of that smart contract for malicious attacks. The incentive behind this being that if it is theoretically possible to buy enough compute force to enforce transactions on the Optimistic network, its highly likely that someone will try. If in theory ten million dollars worth of Ether was stored in a contract , financially speaking, it makes sense to spend a fraction of that number on purchasing computational power. This general idea gives rise to the argument that Optimism is not decentralised enough to maintain its security model indefinitely.

3.5.2 An Honest Assumption

The foundational rule that the Optimistic network functions by is that in any given set of validator nodes, 1-of-N nodes will be honest and always provide the full merkle state of the network for validation of transactions. Although this proves true in practice , it is theoretically possible to coerce node owners on the network to perform a 51 percent attack or to pool compute resources to validate invalid transactions and roll back blocks to allow for double spending. Although these possibilities are implausible they are not impossible. Some of these issue are addressed with the move to Proof of stake with ETH 2.0. Stake holders will be even less incetivised to risk their staked either within a node to partake in malicious attacks on the networks security. This proposes two realities in which Optimisms security model exists. In an Exmaple of Alice , she may trust in the 1-of-N rule and the optimistic rollup becomes a trusted system. In this state the rollup works at a scale and is functionally viable in the sector. In the other reality Alice takes a trustless approach and decides to validate the state of her Optimistic rollup herself, this is an issue as computational costs scale linearly with optimism optimisation of Ethereum. In a scenario where Optimism scales Ethereum 20x, the computational cost to self validate the rollup will also scale 20x. This practice becomes one of an altruistic nature , and possibly a point of failure in Optimism's decentralisation. This in itself is a potential security risk in the future.

3.5.3 Conclusions

In conclusion the Optimism network shows great promise in the Layer 2 space allowing for general purpose smart contract deployment fluidly between the EVM and OVM, this gives it a larger chance at mass adoption by the community and its DApp development Eco-system. It uses the solidity smart contract language giving developers ease-of-access to its development capabilities from the start. These tools give Optimism great presence

in the Layer 2 space however its security risks although not very feasible in reality are still theoretically possible, some more then others. In comparison to ZK-rollups, which have made massive development bounds even since the point at which this project began, they are the lesser secure Layer 2 rollup. It is the general community opinion that in the given state most developments will move to ZK-Rollups once they have achieved equal interoperability as Optimism has with EVM and OVM. The playing field is quite competitive and the future is not set in stone but in the current climate it does look like ZK-Rollups will be taking the lead soon.

Chapter 4

Implementation

As an evaluation of the Optimistic rollup, a dive into the development space was undertaken. Although the Optimistic network is in theory the most interoperable and easy to port to rollup network currently available, it did not come as an easy task in any regard. The initial aim of this section was to deploy and build a DApp to the Optimistic network to avail of the increased transaction speeds and lower gas costs. The DApp being a crowdfunding service similar to "goFundMe" or "Git Grants". This however proved out of scope within the time frame so a simpler approach of deploying a smart contract to the optimistic network was attempted to compare both TPS and gas fees. This also however proved impossible due to specific issues encountered and described later in the section. This section will describe the intended work done as well as the issue encountered which prevented development in the space. This evaluation can be broken into the following sections of research conducted:

- Software
- Compilers
- Issues faced
- Workarounds

4.1 Software

In the area of Blockchain in general software is a fickle creature. Although leaps and bounds in the development of the Blockchain itself have occurred, the software somewhat lags behind. *"Solidity is a contract-oriented, high-level language for implementing smart contracts. It was influenced by C++, Python and JavaScript and is designed to target*

the Ethereum Virtual Machine (EVM)" [23]. As Solidity was built with the intention of targeted use with the EVM it does its job quite well and as its basis is in JavaScript and C++, it is quite intuitive to pick up. However, where the real issue lies is within the development and compiler software. As optimism is a new development in the Ethereum space its resources and documentation are somewhat lacking. Along with its early community uptake in the space, very few if not no public examples of Optimistic DApp deployment are available to a general developer like myself. This made creation in the space incredibly difficult as many of the facilities required to develop were locked behind whitelists or simply an inability to gather the resources required to test. For example, much like the Ethereum network, Optimism has a public test-net where theoretically a person can deploy there Optimistic Application and test it against a live network. This in practice was not possible for the following reasons:

- The Optimistic Kovan test-net requires Optimistic Kovan Ether(Optimistic KEth) in order to pay for deployment much like any other Ethereum network. To obtain Optimistic KEth one must first have access to Kovan Ether(KEth), Ethereum's own Kovan test net on Layer 1, and use the Optimistic gateway to convert from KEth to Optimistic KEth. Simple in theory but in practice impossible. To obtain KEth today you must request it from a KEth faucet. These are publicly visible wallets containing KEth that will deposit a set amount of KEth upon request. The issue at hand being that none of the faucets work at the current time making KEth impossible to obtain. This removed Optimistic Kovan as a potential testing and deployment space for this project.

- The Optimistic Mainnet although up and running on Infura as well as Optimism's own mainnet has its own set of issues. Although a contract call and deployment hash can be found on Etherscan.io shown in figure 4.2, a site for public information about any transaction on the network, for this project's attempted deployments to the network, they are rejected by the network for an unknown reason. This was unresolved as there is not a large amount of community knowledge in regards to deployment errors due to the early stages of deployment Optimism is in currently. It is mentioned on Optimism's documentation that the Optimistic mainnet would be on a "whitelist" for deployments to the network while in its development stages. This however was not the case for the Infura Optimistic mainnet and yet both networks returned the same deployment error shown in figure 4.1.

As these issues continued to to arise with any form of public net for Optimism the pathway taken for testing changed towards running a localised OVM implementation on my local linux machine. This process involved the use of Docker, a software for virtualising

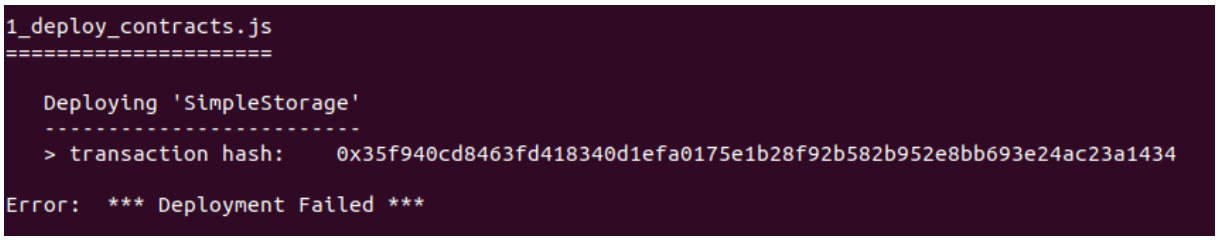

Figure 4.1: Figure of error experienced within the optimistic mainnet through deployment

containers, to virtualise the OVM inside of the EVM. More success was found on this pathway as already existing software packages existed in their early development to help developers start in the space. This implementation required two different packages:

- Optimism box developed by the trufflesuite found here: *https://github.com/truffle-box/optimism-box*
- The Documentation and Optimism monorepo repository for the Optimism local node found here: *https://github.com/ethereum-optimism/optimism*

To deploy a contract to the localised OVM, the smart contract code must first be compiled using the OVM compiler, this being the functions discussed previously making use of the Transpiler 3.2.4. After the contracts are compiled successfully the Docker image of the OVM must be started through the **npm run startLocalOptimism** terminal command. Although unclear as to why, from experience the Docker image will always exit with an error if the **npm run compile:ovm** command has not be run prior to its launch. Once the docker image is built the local Optimism implementation will be deployed onto the localhost:127.0.0.1:7545 although this can be changed within the *truffle-config.ovm.js* file. From here deployments from the terminal are possible and comparisons between a local Ethereum deployment and a local Optimism deployment can be found in section 5.

4.2 Compilers

In the development space currently there are two main development packages, those being *Truffle* and *Hardhat*. Each comes with a set of benefits and drawbacks which will be explored here. The truffle compiler has a long standing Ethereum development history being one of the first and foremost compilers to arise. This gives the compiler a long standing community backing as a plethora of previous errors, solutions and general discussions around its working can be found within the community forums. This was the deciding factor in the decision to go forward with truffle rather then Hardhat. Hardhat

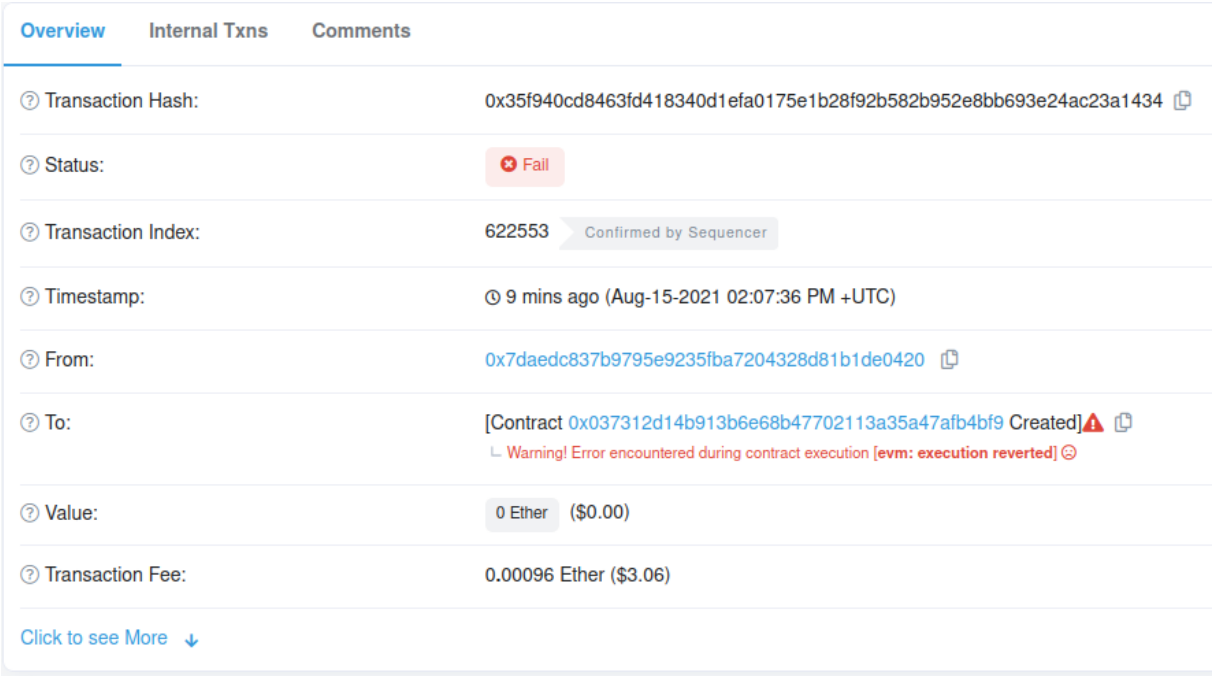

Figure 4.2: Figure of error experienced from public transaction data obtained through optimistic Etherscan

is a relatively new development package and although it shows great efficiency benefits over truffle as well as some welcome simplifications to the deployment process, it lacks that community knowledge base that Truffle has making any debugging or error checking a much more laboursome task. Hardhat currently is being adopted on a large scale by the community and it is only a matter of time before it rivals truffle for a knowledge base and Documentation.

4.3 Issues faced

The issues faced within this project caused many alterations in the original scope of this project. The inability to deploy to a live network inhibited the ability to test live functionality with smart contracts as although the local implementation of the Optimistic network allowed for deployment, through testing throughout the project, it was found that calls to a contract on the local Optimism network were unsuccessful as a truffle console command for asynchronous calls to the network only works for the local Ethereum network and not Optimism. This limited the ability to test previously discussed benefits of the network, things such as:

- Transactions per second
- Throughput of the system

- testing of Withdrawal windows and fraud proofs
- Testing the network on ability to handle congested calls to a single contract by means of a looped call to a contract on both networks and monitoring the difference in throughput

Although research into these areas was extensive in previous sections, the aim was to implement a live demonstration of these factors to validate the claims of research done. Although this was ultimately unsuccessful, a large amount of knowledge about the development space was gathered and allowed for possible alternatives or workarounds to be theorised in section 4.4.

A quick note in terms of operating system used, throughout the project issues arouse through the use of Windows 10. This error came with the attempted Deployment of the docker image from the Optimism monorepo to either the windows Bash shell or inside of the Microsoft Visual Studio Code bash terminal. Although all dependencies were met in both cases the Docker image would only deploy to a native Linux machine which is where this project ended up. It is possible that some variable within my own local windows installation caused both instabilities but upon two fresh installs of the operating system the error still persisted. This led to the move to a Linux system, name UBUNTU 20.0.4. This proved successful.

4.4 Possible Solutions

Although the time frame of this project as well as obstacles that prevented development impacted the testing done, many possible solutions were discovered without time to implement them within the time frame. An example of this being the inability to obtain KEth. This problem is most likely solved with just more time as the faucets will be fixed by the owners, such as Metamask, and allow the transfer of Kovan Ether to Optimistic Ether once again for testing. As the network issue is theoretically fixed this by extension resolves the issue surrounding testing on the local Optimism network. With a live net to test on a localised Optimism becomes the inferior option as live net Applications are interact-able trough online IDE's such as **Remix** hosted at *https://remix.ethereum.org/*. This IDE has in built smart contract interaction through the use of function buttons after deployment to a network. This however is limited right now by the slow uptake of the Optimistic network and its deployment differences based on its higher required gas fee for moving from Layer 1 to Layer 2. This is reflected in the total Ethereum prices seen in Abstract figure 2 and Figure 3

In the line of testing locally it is possible to connect a smart contract deployed to a local net to a webpage through the use of the Web3 library. This library is the most common for this function on Ethereum mainnet Dapp's. As an implementation of this was outside of the time frame it was not tested but in theory and as far as can be found by ways of Documentation, this method would allow for the localised testing that was missing from this project.

Chapter 5

Evaluation

From the tests that were possible to carry out on the network and knowledge gathered throughout the course of the project a general evaluation of Optimism and the eco-system that it finds itself within begins to show. Optimism currently plays the larger roll in terms of functionality within the rollup development space, its interoperability with the EVM gives it an advantage in terms of mass adoption against its counterpart the ZK-rollup. However this does not appear to hold in the future. Optimism as it stands will fall behind in terms of TPS, scalability and security to ZK-rollups as the development pace at which it is improving has as of August 15th 2021 already almost caught up to the interoperability of the OVM.

Although it will potentially fall behind ZK-rollups in terms of adoption at a later stage, Optimism still shows promise in the area of its OVM becoming a validity checker for other layer 2 solutions such as sidechains and Plasma. This would give Optimism a purpose past its layer 2 rollups and give its system another primary use case within the Ethereum Eco-system.

In terms of security while its security model does prove strong it does raise concerns that ZK-rollup do not in terms of semi-centralisation around the locked Ether stored in a single Layer 1 Optimistic contract, creating somewhat of a Honey Pot situation and incentivising the potential large cost purchasing of compute power on the network to circumvent the security protocols silently and illicit invalid transactions and double spending. Although only a theoretical worst-case scenario it still requires attention.

Optimism's potential scaling is yet to be seen , growing to an approximate 2000+ TPS with an implementation of BLS signature aggregation model, and opens the possibility of it winning mass adoption just through the course of "first come first served". If Optimism has mitigated the gas and transaction issues seen in Ethereum before Zk-rollups are up to par then its not outside the realm of possibility that the community continues with the

network it knows rather then the underdog it doesn't. With ETH 2.0 and its Sharding and beacon chain model this number could grow drastically giving more weight to this potential future for Optimism.

Optimism's largest drawback is its withdrawal times. In comparison to Zk-rollups current 20 minute average withdrawal time from layer 2 to Layer 1, the one week waiting period for Optimistic rollups seems quite outrageous. Although this will be circumvented by liquidity provider , which will in turn make the user experience more or less the same across both rollups, the cost of liquidity withdrawal will more then likely be quite pricey compared to any that develop on the ZK-rollup chain.

Chapter 6

Conclusions & Future Work

6.1 Future Work

As stated previously with the re-evaluation of the scope of this project came the simplification of the original idea. The future work for this project would be to implement the original proposed idea of a Community funding site based on the Optimism network. In practice this would work through the use of a Web3 library in conjunction with the OVM compatible smart contracts written in solidity. As simplified versions of this application have been created on Ethereum's mainnet as well as large scale crowdfunding like "Git grants", it seems an appropriate application for the Optimistic network for the following reasons:

- As people bring money to a community funding site to donate and deposit to a certain cause , in most cases a user has a pre-specified amount of funds they wish to donate. With this outlook, the locking of funds to ETH layer 1 does not impact the general functionality of the application as many do not intend to withdraw funds back to layer 1.

- With deployment to the Optimism network comes all the throughput benefits as well as improved transaction speeds aiding the user experience while on the application

- Optimism's main drawback is its withdrawal times , although as discussed this is possibly mitigated through the liquidity providers or markets, in this case a withdrawal window will only occur in the finalisation of a community funding project. This means that only the project being funded will wait through the 7 day Dispute time Delay Window which in practice should not matter due to the amount of time already set by the project for how long a fund should last, mainly lasting weeks.

I believe this would be a good fit for the Optimistic network for reasons stated above and in the future would prove successful in the space based off of previous examples of this type of application succeeding

6.2 Conclusion

Throughout this project I explored the areas of Blockchain, Ethereum, Layer 2 solutions and the Optimism network. With an in-depth look into both Optimistic rollups and ZK-rollups many conclusions can be drawn. Layer 2 solutions as a whole can be seen as a massive benefit to the Ethereum network allowing for both the temporary scaling it needs to circumvent its current bottlenecks while awaiting the final implementation of ETH 2.0. Along with ETH 2.0 comes the ability to scale these Layer 2 solutions even further as their ability to handle data only improves in this new landscape.

In the battle between ZK-rollups and Optimistic rollups it seems apparent that, although currently Optimism is in the lead in terms of actual implementation in services such as *Uniswap*, the token swap platform for Ethereum, it will eventually fall behind ZK-rollups as even now they are close to a full iteroperability between the Zk-rollup and the EVM. Although not as seamless as Optimism claims to be at present, it does not look like it will take very long to overshadow their counterpart. Although obviously it is possible that future developments will alter this course it is also arguable that Optimism is beneficial for Zk-rollups right now as they provide a stop gap while ZK-rollup continues to develop to the optimal point at which it can take over as the predominant rollup protocol.

This Layer 2 space is exciting currently and although testing of the space was incomplete, it shows huge potential and a large impact on the way Ethereum will function in the near future as we await ETH 2.0 and Proof-of-stake. As continuous development occurs it is also possible that another contender for the title of top-rollup will arrive. This is not unlikely as already cross implementations of multiple layer 2 solutions used in parallel are being tested and experimented with. As Ethereum continues to grow in efficiency and functionality so too will its Layer 2 solutions. The upcoming period of development is an exciting one and many great things for this space lie on the horizon.

Bibliography

[1] Alexey Akhunov, Eli Sasson, Tom Brand, Louis Guthmann, and Avihu Levy. Eip-2028: Transaction data gas cost reduction, 2019.

[2] Bitcoin-Wiki. Multi-signature, 2021.

[3] Dan Boneh, Manu Drijvers, and Gregory Neven. Compact multi-signatures for smaller blockchains. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018*, pages 435–464, Cham, 2018. Springer International Publishing.

[4] William Entriken. Layer 2 rollups, 2021.

[5] William Entriken. Plasma, 2021.

[6] Optimism Ethereum. Ovm deep dive, 2020.

[7] Ethereum.org. Ethereum virtual machine (evm), 2021.

[8] Ethereum.org. Security changes with the incoming of eth 2.0, 2021.

[9] Ethereum.org. Shardchains, 2021.

[10] Karl Floersch. Ethereum smart contracts in l2: Optimistic rollup, 2019.

[11] Alex Gluchowski. Optimistic vs. zk rollup: Deep dive, 2019.

[12] Alex Gluchowski and Alex Vlasov. Introducing matter testnet, 2019.

[13] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016*, pages 305–326, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[14] IvanOnTech. Exploring optimism's optimistic virtual machine (ovm), 2021.

[15] Georgios Konstantopoulos. How does optimism's rollup really work?, 2021.

[16] Mark. Ethereum virtual machine explained, 2018.

[17] Monolith. Understanding defi: Layer 2 explained, 2020.

[18] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2009.

[19] Ethereum Optimism. Introducing the ovm, 2019.

[20] Artem Payvin. The data availability problem, 2019.

[21] Sarwar Sayeed and Hector Marco-Gisbert. Assessing blockchain consensus and security mechanisms against the 51 *Applied Sciences*, 9(9), 2019.

[22] Tobias Schaffner. Scaling public blockchains. *Applied Sciences*, 2021.

[23] Solidity. Solidity, 2020.

[24] Starkware. The optimistic rollup dilemma, 2020.

[25] G Wood. Ethereum yellow paper: a formal specification of ethereum, a programmable blockchain. *Accessed on: Mar*, 6, 2019.

[26] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. Rapidchain: Scaling blockchain via full sharding. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 931–948, New York, NY, USA, 2018. Association for Computing Machinery.

Abstract

- fig 1 source: https://community.optimism.io/docs/protocol/protocol.html
- fig 2 source: own local system
- fig 3 source: own local system
- fig 2.1 source: http://www.ccs.neu.edu/home/noubir/Courses/CSG252/F05/slides1.pdf
- fig 2.2 source: https://www.semanticscholar.org/paper/ An-implementation-of-enhanced-public-key-Tan-Yau/ddeb5fe64111f9ed4bfb23a3c919363cb57e7993/figure/0
- fig 2.3 source: https://commons.wikimedia.org/wiki/File:Web_of_Trust_2.svg
- fig 2.4 source: https://blog.csdn.net/http188188/article/details/87883330
- fig 2.5 source: https://etherscan.io/chartsync/chaindefault
- fig 2.6 source: https://www.researchgate.net/publication/333824348/figure/fig4/AS:770773534924800@1560778139261/Blockchain_Merkle_tree_root.jpg
- fig 2.7 source: https://academy.bit2me.com/wp-content/uploads/2019/05/cueva-de-alibaba-800x220.jpg
- fig 3.1 source: Own illustration
- fig 3.2 source: https://miro.medium.com/max/2000/1*NX-HxmrKRG3v09JtCMT1pg.gif
- fig 3.3 source: https://miro.medium.com/max/2000/1*NX-HxmrKRG3v09JtCMT1pg.gif
- fig 3.4 source: https://miro.medium.com/max/2000/1*txRaipmV1qaxFNRriluFLg.gif
- fig 3.5 source: https://wwz.unibas.ch/fileadmin/user_upload/wwz/00_Professuren/Schaer_DLTFintech/Lehre/Tobias_Schaffner_Masterthesis.pdf
- fig 3.6 source: https://wwz.unibas.ch/fileadmin/user_upload/wwz/00_Professuren/Schaer_DLTFintech/Lehre/Tobias_Schaffner_Masterthesis.pdf

- fig 4.1 source: own local system
- fig 4.2 source: https://optimistic.etherscan.io/tx/0x35f940cd8463fd418340d1efa0175e1b28f92b582b952e8bb693e24ac23a1434

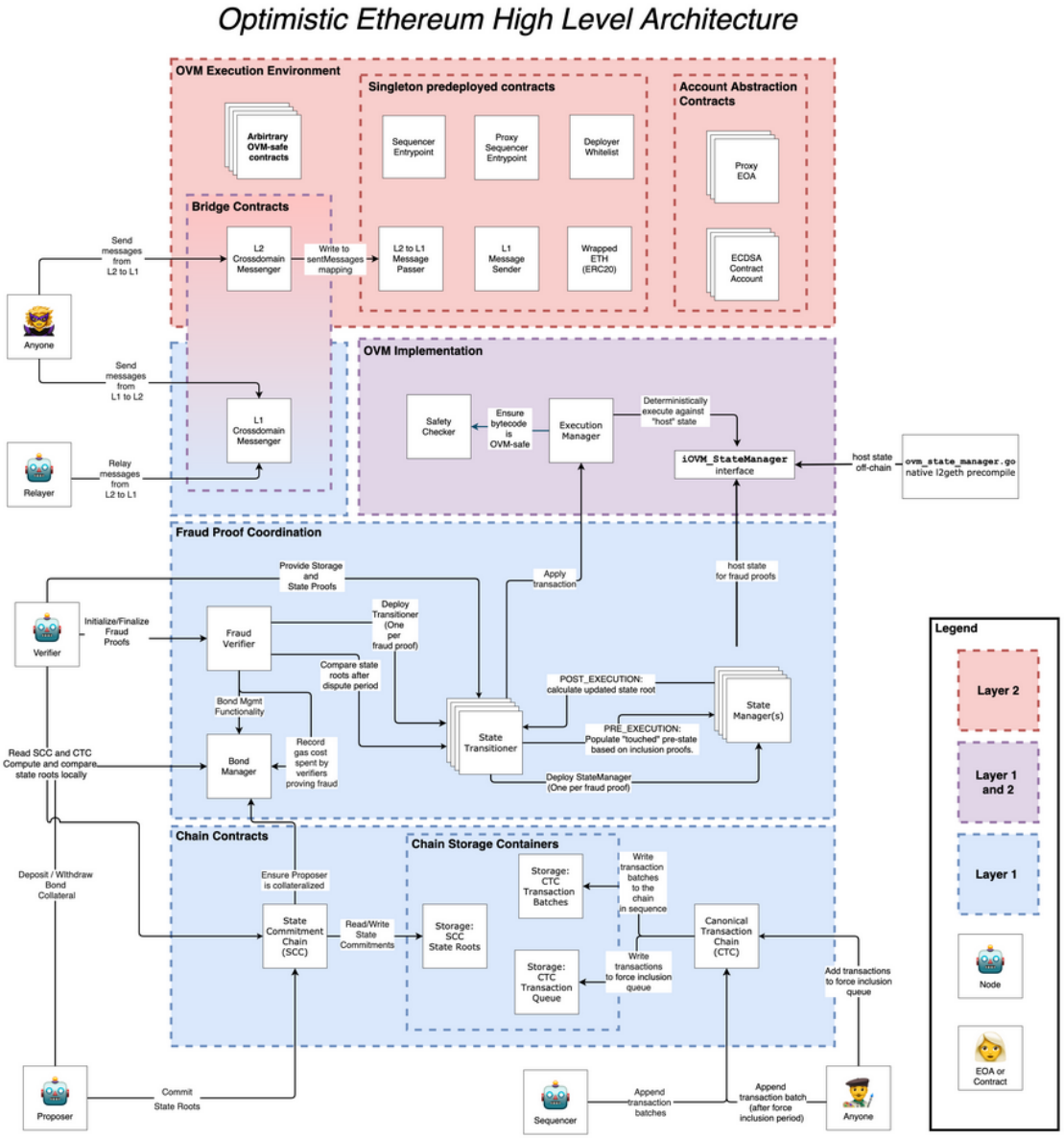

Figure 1: High level diagram of the OVM

```
matt@matt-desktop:~/Documents/Project$ npm run migrate:ovm --network=optimistic_ethereum

> optimism-box@1.0.0 migrate:ovm
> truffle migrate --skip-dry-run --config truffle-config.ovm.js --network $npm_config_network


Compiling your contracts...
===========================
> Everything is up to date, there is nothing to compile.



Starting migrations...
======================
> Network name:    'optimistic_ethereum'
> Network id:      420
> Block gas limit: 11000000 (0xa7d8c0)


1_deploy_contracts.js
=====================

   Replacing 'SimpleStorage'
   -------------------------
   > transaction hash:    0xc063cdc6b88487b9e847154da8168ed6bca07f3496bd1f10e46f70d1b28aa962
   > Blocks: 0            Seconds: 0
   > contract address:    0x5FbDB2315678afecb367f032d93F642f64180aa3
   > block number:        21
   > block timestamp:     1628699565
   > account:             0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266
   > balance:             4999.9034278
   > gas used:            1569481 (0x17f2c9)
   > gas price:           0.015 gwei
   > value sent:          0 ETH
   > total cost:          0.000023542215 ETH

   > Saving artifacts
   -------------------------------------
   > Total cost:      0.000023542215 ETH


Summary
=======
> Total deployments:   1
> Final cost:          0.000023542215 ETH
```

Figure 2: Deployment of contract to Local Optimism

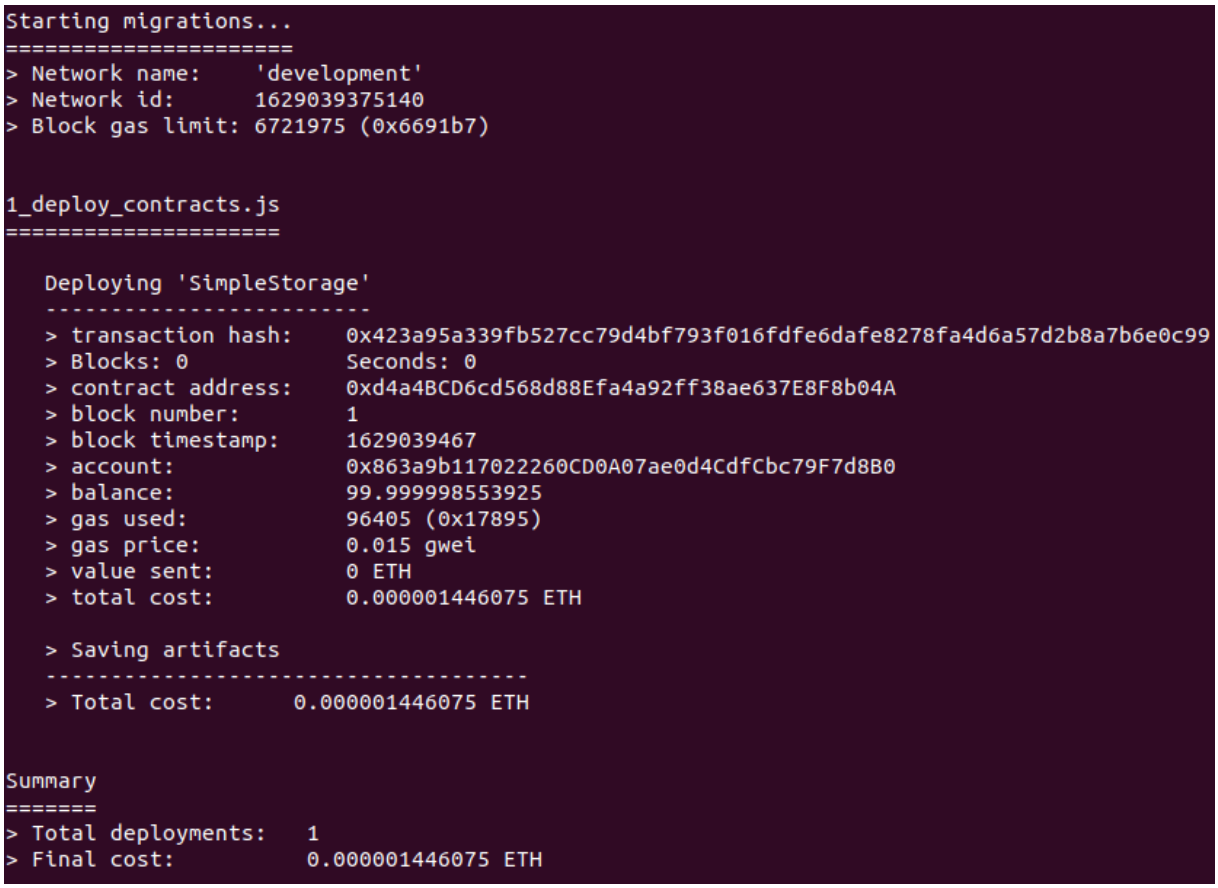

Figure 3: Deployment of contract to Local Ethereum