Master's Thesis

Scaling Public Blockchains

A comprehensive analysis of optimistic and zero-knowledge rollups

Tobias Schaffner
tobias.schaffner@stud.unibas.ch
2016-065-443

January 14, 2021

Supervised by:
Prof. Dr. Fabian Schär
Credit Suisse Asset Management (Switzerland) Professor for
Distributed Ledger Technologies and Fintech
Center for Innovative Finance, University of Basel

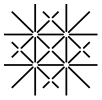
Abstract

Increasing the transactional throughput of public blockchains is the primary focus of blockchain research today. Achieving this without compromising security or decentralization is the holy grail and will be pivotal for many broader economic use cases on public blockchains.
This thesis provides a structured overview of potential scaling solutions before thoroughly introducing and comparing zero-knowledge rollups and optimistic rollups. Both are promising layer 2 solutions that claim to scale public blockchains significantly in the near future. Furthermore, this thesis introduces fundamental concepts such as zero-knowledge proofs and examines potential attacks based on game-theoretic principles.

Keywords: Blockchain, Ethereum, Scalability, Zero-knowledge Proofs.
JEL: C25, C70, D82, E49, G19

Contents

List of Figures



University of Basel

Center for
Innovative Finance

Plagiarism Declaration

With my signature, I certify that the information I have given about the tools I have used in writing my thesis and about the help I have been provided is true and complete in every aspect. I have read the information sheet on plagiarism and fraud, issued on 22 February 2011, and am aware of the consequences of such action.

Tobias Schaffner

Basel, January 14, 2021

1 Introduction

Public blockchains and the blockchain technology in general enjoyed a tremendous rise in popularity and gained a lot of attention since the beginning of the last decade. Initially, most public blockchains were based around financial use cases such as decentralized cryptocurrencies, and the beneficial properties of the blockchain technology itself were commonly overlooked. Over the last few years, the broader economic community slowly started to realize what benefits public blockchains in combination with technologies such as smart contracts could actually bring. Today, the underlying blockchain technology is perceived as the real disruptor that may revolutionize many industries at the same time. Yet, the lack of scalability remains an incisive issue of most public blockchains that prevents a wide range of promising use cases from being realized. In fact, scalability has emerged as the primary issue that needs to be resolved in order to achieve mass-adoption, especially considering the increasing popularity of permissionless blockchains in recent years.

In this thesis, I will provide a structured overview of various scaling solutions that address scalability concerns of permissionless blockchains and comprehensively introduce two very recent approaches. Both of these solutions pledge to be the most promising and both appear likely to solve public blockchains' lack of scalability in the near future. Therefore, I am going to assess, analyze and compare the two approaches called *optimistic rollups* and *zero-knowledge rollups*.

As a preliminary remark, this thesis has a strong focus on Ethereum. All solutions which are introduced within this thesis are based on the Ethereum blockchain. From now on, if not noted differently, I will use the terms *Ethereum blockchain* and *public blockchain* interchangeably. However, throughout this thesis, Ethereum will also be benchmarked to Bitcoin and other less established alternatives. Although the Bitcoin blockchain may possibly be the best known blockchain, Ethereum is considered to be the most relevant public blockchain. This is because of its economic interest, its daily on-chain traffic and especially its ability to employ turing-complete smart contracts.

2 The Scalability Problem

The lack of scalability is probably the core issue that most established public blockchains like Ethereum or Bitcoin struggle with. In fact, scalability has been a concern since the first day of public blockchains. When Satoshi Nakamoto published the Bitcoin whitepaper [Nak08] and single-handedly invented public blockchains in the end of October 2008, one of the first public reactions to Nakamotos's whitepaper came from James A. Donald on November 2 and stated: «We very, very much need such a system, but the way I understand your proposal, it does not seem to scale to the required size [...]»[Opa18]. Of course, these scalability concerns were of a different nature, but even today, more than twelve years on, the public blockchain's lack of scalability still persists.

The main reason for this issue is the underlying network protocol which requires that every transaction is processed by every node in the network. Both, Bitcoin and Ethereum, are currently using a *proof-of-work* based distributed consensus mechanism that slows down the block creation significantly. 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. As a consequence of this consensus mechanism, every node in the network has to verify these efforts by the miners and keep a copy of the current state of the blockchain. In Ethereum, such new block creations happen every 10 to 20 seconds, whereas in Bitcoin a new block is added to the blockchain only every 10 minutes. This drastically limits the transactional throughput of both of these public blockchains. If you break down the number of transactions within one block on the Ethereum blockchain, this leads to only around 12 to 25 transaction per second on average. [CDE+16] [WZS19]

2.1 Scalability benchmarks

In contrast, state-of-the-art centralized global payment systems like Visa are able to scale up to more than 65'000 transactions per second at peak [VIS19]. Despite these impressive numbers, it should be noted that such figures resulted from stress tests conducted by Visa engineers and not on a daily basis. However, outside of such testing environments, the VisaNet still handles about 2000 transactions per second on average, with peak rates up to 4000 transactions per seconds [Mou16].

Exactly the same can be said about many newer and less established public blockchains like *EOS*, *BitcoinCash* or *Qtum*, to name just a few. Many of these alternatives were developed only a few years ago and some of them even emerged from a hard fork of Bitcoin. The creators of these newer blockchains had the chance to react to certain shortcomings of the more established ones. As a result, some of these less established public blockchains allegedly claim to scale up to several thousand transactions per second, which is often coupled with trading off some level of decentralization or security in return. In fact, when it comes to scalability there are some promising alternatives to Bitcoin or Ethereum. However, similarly to the VisaNet, most of them can only theoretically scale up to such large numbers and realistically there is no demand for such high amounts of throughputs on these platforms.

This fact is beautifully illustrated on the txstreet.com website, where buses represent new blocks and passengers represent incoming transactions. The sizes of those buses are congruent with the block sizes of the corresponding public blockchain. On one side of the street you see small Ethereum buses leaving very frequently and completely packed. On the other side you see large BitcoinCash buses leaving very irregularly, about every 10 minutes, with only very few passengers in total. In addition to this, it also shows that there is a large queue of passengers (transactions) willing to pay a lot of transaction fees to get into one of the small Ethereum buses (blocks) as quickly as possible. [CDE+16] [O'N19] [WZS19]

2.2 Scalability trilemma

The scalability trilemma describes a trade-off relationship between the following three desirable properties of a public blockchain architecture: *Decentralization*, *Security* and *Scalability*.

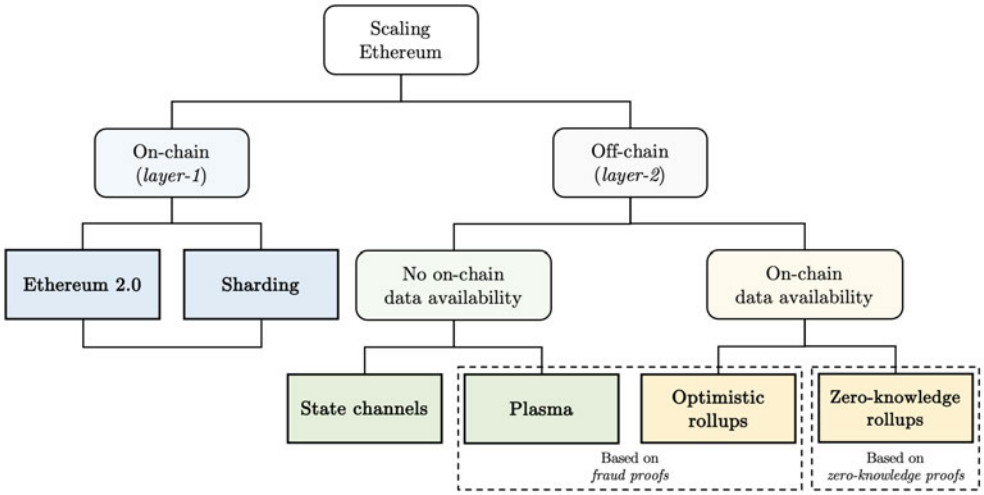
The trilemma claims that it is not possible to equally maximize all three of these properties at the same time. Satisfying two of them will always come at the expense of disregarding the third. It is called scalability trilemma, because established public blockchains typically struggle to scale sufficiently. This becomes very clear when considering the trilemma in the context of the Ethereum blockchain. While this specific blockchain is very secure and perfectly decentralized, its scalability definitely falls behind.

As described above, scalability is an essential ingredient for public blockchains in order to achieve mass-adoption and to attract mainstream applications being deployed on-chain. On the other side, security is obviously inevitable and a necessity for any operating system or network. Therefore, decentralization may appear as the least important of the three. However, a decentralized architecture with thousands of nodes is actually the core ingredient of a public blockchain and responsible for all the benefits in the first place. Of course, all of these attributes are not a priori binary and can be mutually aligned to serve a particular use case.

As a matter of fact, this trilemma leads to many blockchains with different applications, by choosing to focus according to their needs. For example, EOS is one of the afore-mentioned newer public blockchains, that scales very good while being secure at the same time. EOS' weakness, however, is their lack of decentralization, especially in comparison to Ethereum. This trilemma can also be taken to the extreme when applying it to a centralized (non-blockchain) architecture. The VisaNet, which we used as a scalability benchmark above, is completely centralized and therefore scales almost infinitely while also being secure at the same time.

Vitalik Buterin, the founder of the Ethereum blockchain, claimed that the holy grail of developing a blockchain platform that maximizes all three attributes is basically unachievable. Therefore, my approach is to focus on Ethereum, a well-established public blockchain that already exploits two attributes perfectly, and to evaluate solutions that may lead to a significant leap in scalability, without sacrificing too much security and decentralization. [Low20] [QG19] [VS18] [WZS19]

3 Scaling Ethereum



Source. Own illustration based on [Tok20].

Figure 1: Overview of Ethereum scaling solutions

Within the Ethereum community there is a common understanding that scalability is one of the last missing pieces before its blockchain achieves mass-adoption among both, private and corporate users. Therefore, finding a feasible scaling solution is currently one of the most active research areas and can roughly be split in two main categories: *Layer 1* and *Layer 2 solutions*.

In this thesis, my research focus will lay on a particular group of *semi-layer 2 solutions* called *rollups*. Nevertheless, for the purpose of completeness and general understanding, I am also going to provide a brief introduction to the most common layer 1 and layer 2 scaling solutions.

3.1 Layer 1 solutions

Layer 1 is a synonym for the underlying blockchain architecture – the main layer. Layer 1 solutions are directly implemented on the main blockchain itself and therefore also called *on-chain solutions*. In fact, they change parameters directly within the core Ethereum protocol, and since

this requires general consensus of the miners, they are often implemented in a rather lengthy process.

3.1.1 Consensus protocol switch

Probably the most obvious on-chain solution is the long-awaited switch from the current *proof-of-work* to the less wasteful *proof-of-stake* consensus mechanism. Despite not only being a sole scaling solution but rather an upgrade of the whole Ethereum environment, I consider this a layer 1 implementation, that also improves scalability. In contrast to the proof-of-work protocol, where consensus is reached by successfully solving computationally hard puzzles, in proof-of-stake a block creator is chosen deterministically by an algorithm based on their own financial stakes. The block creator receives only transaction fees instead of a block reward, that a successful proof-of-work miner would be rewarded with.

This change of consensus protocol is not only going to improve Ethereum's transaction speed and throughput, but also going to improve the efficiency by shifting away from the very resource-intensive mining of the proof-of-work mechanism. This new environment will be implemented with the serenity update and is conventionally called *Ethereum 2.0*. Besides the switch from proof-of-work to proof-of-stake, it will also enable another scaling solution called *sharding* and introduce a next generation EVM called *eWASM*. [QG19] [Wal20] [WZS19]

3.1.2 Sharding

A well-known layer 1 solution which could be enabled by the above-mentioned switch to Ethereum 2.0 is called *sharding*. The underlying concept of sharding has already been used to improve the scalability of other areas in the past, such as relational databases and big data platforms. In the current blockchain environment all blocks and transactions are validated by every node within the whole network. Sharding intends to group these network nodes horizontally into different smaller units,

called *shards*. Each of these shards will then reach its own consensus and create a subset of a new block that will be added to the main blockchain. This has the benefit that nodes no longer have to store and process the entire transactional load of the network. Therefore, individual block creation can become much more efficient.

However, the application of sharding within the Ethereum blockchain is still considered very complex. It would take a very careful design of a sharding mechanism to ensure that all fragments are sufficiently decentralized, and individual shards are not owned by a single entity. Another challenge is the inter-shard communication, that may lead to an even less efficient blockchain after all. The concept of sharding is somehow related to a layer 2 solution called *plasma*, that I will briefly introduce in section 3.2.2. [QG19] [Wal20] [WZS19]

3.1.3 Conclusion

Despite some remaining challenges, sharding, coupled with the switch to a proof-of-stake consensus mechanism, could solve many of Ethereum's problems, including its lack of scalability. Since both of them are layer 1 solutions and somehow dependent on each other, they have the advantage that everything could be implemented on the existing blockchain, which would keep the architecture and the usability rather simple. If these layer 1 solutions can be implemented properly, they are definitely promising and efficient approaches to solving the scalability problem.

Yet, such fundamental changes directly on the main blockchain itself are usually not easy to implement. In fact, these solutions modify the underlying parameters of an operating public blockchain that is up and running. Such adjustments are typically very slow because they have to be implemented very carefully with several forks in the process, which naturally involves some risk. A successful hard fork is only possible with a general consensus among the miners within the network.

At the time of writing, Ethereum was in the middle of this ongoing transition called *serenity update* from the current environment to Ethereum 2.0. Although some hard forks have already taken place, it is still difficult to predict the exact timeframe for when the transition to proof-of-stake (among many other improvements) will be officially completed.

Since even very small changes on the mainchain are very hard and slow to implement, we will now take a look at layer 2 solutions that provide a faster and easier implementation for Ethereum. Due to the fact that a fully functional Ethereum 2.0 network is probably still a few years away, layer 2 solutions may already bridge the gap towards more scalability today. [QG19] [Wal20] [WZS19]

3.2 Layer 2 solutions

Layer 2 solutions, also called *off-chain solutions*, are no longer implemented on the main blockchain itself but rather form add-on solutions built on top of the base layer. They intend to take workload away from the blockchain, by shifting the transaction computation off-chain. In this case, the Ethereum blockchain acts as an arbitrator that provides certainty and security, even though the transactions are no longer performed on-chain. This idea opens up a large field of cryptoeconomic systems that can optimize the efficiency and increase the throughput of an underlying blockchain. However, it is still a very important aspect that these multi-layer architectures should not sacrifice neither decentralization nor security. [Fou20]

Again, I will briefly introduce various concepts of several layer 2 solutions before analyzing a particular off-chain solution called *rollups* very comprehensively in sections 4 and 5.

3.2.1 State channels

State channels are basically a two-way interaction channel between two parties. Such state channels can be based on multisig transactions or on smart contracts, where the participants previously agree on the conditions and deposit funds onto the channel. Since these peer-to-peer interactions would otherwise all occur on the main blockchain, such state channels can be very useful to provide a very cheap and fast way to send and receive micro-payments. The main advantage of the state channels, compared to a traditional blockchain transaction, is the instant finality coupled with negligible transaction fees. As soon as the set of interactions between the parties has ended, the final state will be published and added to the blockchain.

A well-known state channel approach is the *Raiden network*. Raiden is in many ways similar to the *Bitcoin lightning network* that enables micropayment channels within the Bitcoin environment. Unlike the lightning network, Raiden is tailored for the Ethereum blockchain by enabling the participants more complex channel interactions, including the use of smart contracts. In addition to this, Raiden acts as a payment channel network that can connect many parties directly to each other, using bidirectional token payment channels. This enables multi-party channels between more than just two users. One major drawback for state channels in general is the requirement that the users need to be online all the time, to sign and agree to the state updates. In addition to this, they have to deposit some funds directly into the channel for an extended period of time, which is therefore not available until the channel is closed again. [AQC19] [Mos18] [Wal20] [WZS19] [Yua19]

3.2.2 Plasma sidechains

Sidechains are smaller blockchains that are simply attached to a public blockchain. The idea behind them is to shift transactions away from the mainchain and process them on faster and more efficient sidechains that are less crowded. Traditionally, using such a sidechain also involves

trusting a third-party, which is naturally not an ideal way to scale an initially trustless blockchain.

Therefore, the concept of *plasma* was brought forward by Poon and Buterin in 2017 [PB17]. This framework was introduced to minimize the required trust in operators, which enabled sidechains to become a valid layer 2 scaling solution. Hence, back when it was first announced, plasma was considered to be a very promising approach to solve Ethereum's scalability problems and enable Visa-like transaction throughputs.

Poon and Buterin introduced a concept of fraud proofs, that aims to reduce the power of a sidechain operator. In theory, if fraudulent activities such as double spending within or between different sidechains occur, fraud proofs can be filed and posted to the Ethereum blockchain, and all fraudulent transactions will be rolled back. Again, the main Ethereum blockchain acts as an arbitrator that resolves disputes and therefore provides the shared security to the whole network of sidechains.

Individual plasma chains are bonded to the main Ethereum blockchain via a smart contract and the individual operators will regularly commit the processed transactions to the main layer. In fact, the transaction data will remain within the sidechain and only the block headers will be submitted to the mainchain. This rooting design allows plasma chains to borrow the securities from the parent Ethereum blockchain without having to establish its own consensus. In addition to this, it theoretically enables exponential scalability gains, if implemented properly. [AQC19] [Mos18] [PB17] [RQ20] [WZS19]

However, a major drawback of plasma is its lack of on-chain data availability, commonly known as the *data availability problem*, which I will address in the following section. This leaves plasma chains vulnerable to fraudulent operators and prevents the whole concept of fraud proofs from working as intended in practice.

3.2.3 Data availability problem

As previously described, the introduction of plasma initially intended to enable trustless sidechains. However, storing transaction data completely off-chain creates a specific problem, especially when users want to exit the sidechain and go back to the Ethereum blockchain. This is the so-called *data availability problem*, that poses a real challenge for several layer 2 solutions, including plasma.

More precisely, transactions that are not directly published onto the Ethereum mainnet cannot be reconciled by other Ethereum users. Hence, there is always a risk that some information is lost on the way from the sidechain back to the underlying blockchain – accidentally or deliberately. For instance, a selective censorship [by the operator] of participant's submissions could lead to fraudulent transactions without them noticing. Since this operator only publishes the block header to the mainchain and not the underlying transaction data, nobody could verify the correctness of this sidechain.

As a result of such dishonest operators, many users would want to exit and withdraw their funds from such a fraudulent sidechain back to the Ethereum blockchain. A successful exit, however, can only happen after a challenge period of about one week has expired. Within this period, all honest users would have to publish evidence that proves fraudulent activities of the sidechain operator. These fraud proofs involve posting the entire valid state of the sidechain onto the mainnet, which would then lead to the Ethereum blockchain resolving the dispute and punishing the offending party. In practice, this is arguably impossible, since these plasma chains can reach an arbitrarily large size, that could never be realistically published onto the Ethereum mainnet. On top of that, every user would need to be a full node, storing the whole transaction history of the sidechain, in order to be able to provide such fraud evidences. This issue is conventionally known as the *mass exit problem*. [ABSB18] [AQC19] [Pay19] [PB17] [RQ20] [WZS19]

3.2.4 Conclusion

The main advantage of any layer 2 solution is that they do not have any effects on the core Ethereum protocol and its consensus mechanism. Since they are just add-on solutions on top of the base layer, they are compatible with almost every on-chain solution. In fact, they could not only coexist but even extend some future layer 1 solutions. Thus, they are much easier to implement, without any risk of modifying an operating system and without having to carefully upgrade the basic parameters of the underlying blockchain via hard forks. However, one issue that both, state channels and sidechains, encounter is the data availability problem and the related issue of trust. This problem has hindered both of these solutions to meet the promises of its developers in the first place. Consequently, there has been a shift towards a more recent generation of level-2 scaling solutions, that aims to tackle this data availability problem – the *rollups*. [WZS19] [Yua19]

3.3 Rollups

The concept of rollups first appeared in 2018 when a pseudonym called Barry Whitehat published a GitHub repository named: *roll_up* [Whi18]. Shortly after, Ethereum founder Vitalik Buterin published an improved version of this initial proposal, calling them *zk-rollups* [But18].

The main aspect of a rollup is to keep the whole transaction data always on-chain, while the transaction computation can be shifted off-chain to enable efficiency gains. Therefore, I consider rollups not as a proper layer 2-solution but rather a *semi-layer 2 solution*. Rollups are somewhat related to plasma but claim to be trustless because they resolve the data availability problem by publishing just enough data on-chain. They compress several transactions into a transaction bundle, in order to decrease transaction size and transaction costs and thus increasing the general efficiency. Because rollups store transaction data on-chain, they are able to significantly increase the security guarantees for individual transactors.

These condensed (off-chain) transactions are not validated by Ethereum's consensus protocol but rather secured by different mechanisms to ensure transaction validity. Depending on these approaches, we are either referring to *zero-knowledge rollups*, in section 4, or *optimistic rollups*, in section 5. As both names already suggest, the validity of *zero-knowledge rollups* is based on *zero-knowledge proofs*, while *optimistic rollups* use a model with optimistic assumptions that relies on *fraud proofs*. [RQ20] [WZS19]

In the following sections, I thoroughly introduce the approaches and components of both rollup solutions separately, before comparing their individual benefits and drawbacks in section 6. In some cases, it is reasonable not to address several components, advantages or disadvantages of only one rollup solution in isolation, until I can examine and compare both approaches to each other in this particular subsequent section.

4 Zero-knowledge rollups

Back when the idea of zero-knowledge rollups first appeared in 2018, plasma was hot property and thought to be the future scaling solution for Ethereum. Recall the concept of plasma, that uses fraud proofs to ensure security on a sidechain, where all the transaction data is stored and processed, leading to the data availability problem.

In short, the basic idea of zero-knowledge rollups is related to plasma, yet the construction is rather innovative and complex. Instead of employing fraud proofs like in plasma, zk-rollups use *zero-knowledge proofs* to ensure security. These so-called *zk-SNARKs* have the benefit that they can verify the validity of large transaction bundles and therefore represent everything that is happening off-chain. By using such zero-knowledge proofs, the transaction data within the rollup chain becomes accessible to everybody on the Ethereum blockchain. Therefore, rollups can successfully tackle the afore-mentioned data availability problem. [Wu19]

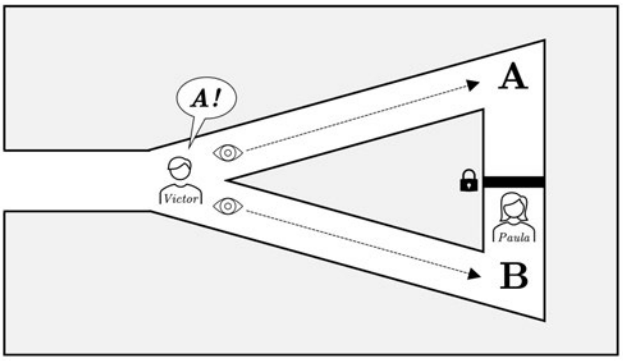
Now, I will first introduce the concept of *zero-knowledge proofs* in detail, because they are the main ingredient and the distinguishing feature at the core of zero-knowledge rollups. Thereafter, I will provide a comprehensive introduction to the construction and implementation of zk-rollups, highlighting some benefits and drawbacks along the way. However, a complete overview of all the advantages and disadvantages will only be possible in section 6, where I can compare zk-rollups to optimistic rollups.

4.1 Zero-knowledge proofs

Although zero-knowledge proofs are a very hot topic today, especially in the context of public blockchains, they date back to the early 1980's, when Goldwasser et al. released a first proposal [GMR85]. Simply put, a zero-knowledge proof describes a situation, where an honest prover can prove his knowledge of a piece of information to a skeptical verifier, without revealing any useful information to the verifier. At first glance,

this may seem like a contradiction, but let's look at an example that illustrates the basic concept very intuitively.

The following example is partly based on the so-called *Strange Cave of Ali Baba* by Quisquater et al. [QQQ+89]. In figure 2 you can see the basic setup of the *zk-cave*. Let Paula be an honest *prover* and Victor be a skeptical *verifier*. The zk-cave has only one entrance and suddenly forks into two passages, A and B. These two passages are connected via a secret tunnel that is locked by a door. This door can only be opened with a secret key, that Paula claims to know. Victor is aware that there is a door but has no clue how to open it. In addition to that, he enters the zk-cave only after Paula is already in the secret tunnel, so that he does not know which passage she took to get there. Victor now asks Paula to come to either location A or B.



Source. Own illustration based on [QQQ+89].

Figure 2: The *zk-cave* from above

Let's assume he chooses location A. Paula opens the door and is able to go to the chosen location A. However, because there is a 50% chance that Paula has already been on that side of the secret tunnel, Victor should not be convinced yet. Victor and Paula will have to repeat this process several (n) times, in order to get to a sufficiently large probability ($1 - 2^{-n}$) which convinces Victor into believing that Paula really knows how to open the door. For instance, after a set of 20 iterations, the counter probability (coincidentally always being at the right end of the

secret tunnel) of $9.54 * 10^{-7}$ is already arbitrarily low. Therefore, Paula has probabilistically proven to Victor, that she really knows how to open the door in the secret tunnel, without revealing any information about how she does it.

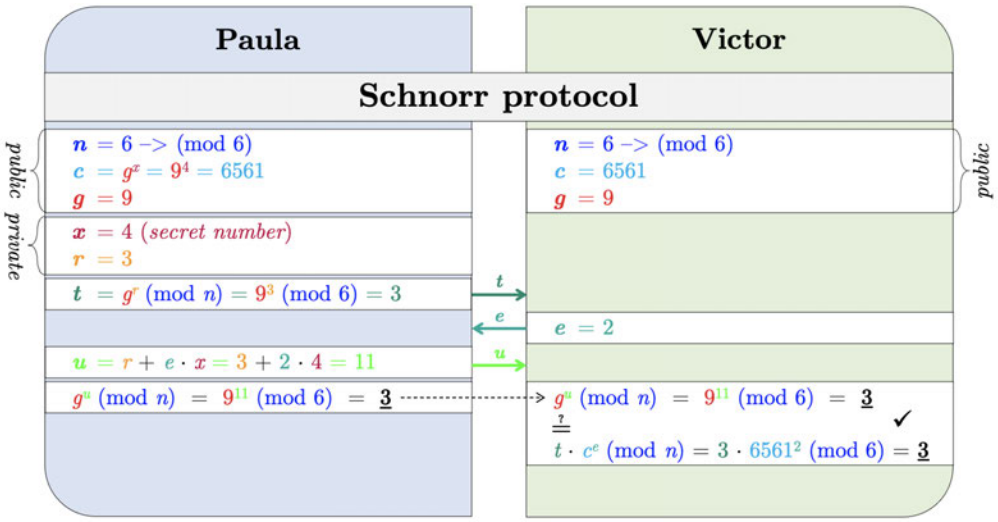
The *zk-cave* experiment above describes an *interactive* zero-knowledge proof, because the verifier and the prover interact with each other and repeat the whole process until the verifier is convinced. An interactive zero-knowledge proof has to meet the following three properties, in order to be considered secure.

Completeness	This property is satisfied if both, verifier and prover, are honest. So, Victor can testify that Paula always went to his chosen location without cheating or colluding with anyone.
Soundness	This implies that it is impossible for a dishonest prover to convince an honest verifier. In our binary case, this soundness assumption can only be fulfilled when Victor and Paula repeat this process several times, so that the counter probability becomes negligible. This repetition is essential, since zero-knowledge proofs are probabilistic and not deterministic.
Zero-knowledge	This property makes zero-knowledge proofs unique. Based on the experiment above, Victor only learns that Paula knows a piece of information but does not learn anything about the information itself. On top of that, a third party, that is spying on them, has no way of checking whether this experiment was genuine or scripted.

Now, I am going to introduce two very common zero-knowledge protocols that should provide an intuition on how such zero-knowledge proofs can be built using fairly simple mathematics. Again, consider Paula to be the prover of a piece of knowledge, in this case a secret number, and Victor to be the verifier. [BBK+13] [GMR85] [QQQ+89] [Yua19]

4.1.1 Schnorr protocol

In 1991, Claus-Peter Schnorr published a protocol [Sch91] that enables interactive zero-knowledge proofs with only three interactions. The *Schnorr protocol* was not the first zero-knowledge protocol, but the most common one today. Like many other cryptographic protocols, the Schnorr protocol is also based on the difficulty of the *RSA problem*, which implies that, given c and n, it is impossible to efficiently compute g and x of an equation $c = g^x \pmod{n}$. In this section, I will introduce the main theoretical principles of the protocol, while also providing a simple overview with a numerical example in figure 3. Please note that the colors aim to provide an easy transition to the numerical example.



Source. Own illustration.

Figure 3: Numerical example of the *Schnorr protocol*

Now, c stands for a committed value and g for the corresponding generator and both values are publicly accessible. The secret number x is individual and only known to Paula. The Schnorr protocol enables Paula to proof her knowledge of a value x that corresponds to the committed value c, without revealing any information about x. Paula can prove this through the following three steps:

(1) Paula randomly chooses a variable r. This enables her to compute $t = g^r \pmod{n}$ and send t to Victor.

(2) Victor now randomly chooses a variable e and sends it to Paula.

(3) Paula then computes $u = r + e * x$ and sends u to Victor.

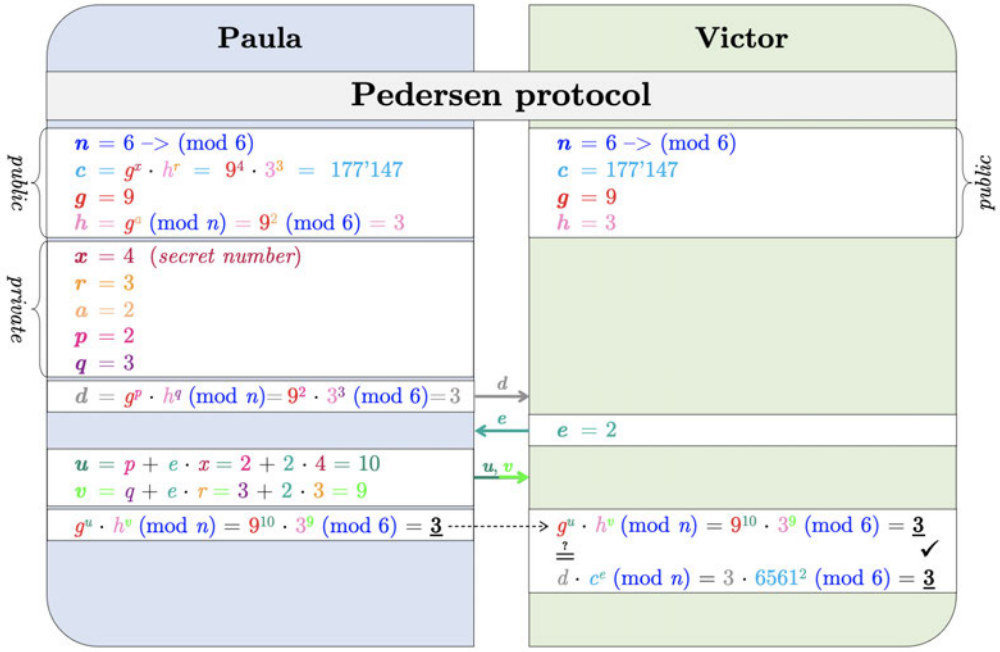
Victor can now compare Paula's outcome of $g^u \pmod{n}$ with his result of $t * c^e \pmod{n}$. If Paula and Victor end up with the same result, Paula has successfully proven her knowledge of x, without revealing anything about it. Note that x could also be called a *private key* with c as its corresponding *public key*. The following mathematical proof in equation 1 shows how this zero-knowledge protocol works and why Victor can trust this protocol. [Fra14] [GLSY04] [Kog19] [Sch91]

$$\begin{aligned} g^u \pmod{n} &= g^{r+e+x} \pmod{n} \\ &= g^r * g^{e+x} \pmod{n} \\ &= g^r * (g^x)^e \pmod{n} \\ &= t * c^e \pmod{n}, \end{aligned} \tag{1}$$

which proves that $g^u \pmod{n}$ equals $t * c^e \pmod{n}$.

4.1.2 Pedersen protocol

The second zero-knowledge protocol that I will go into greater detail was introduced by Torben Pryds Pedersen, also in 1991 [Ped91]. The *Pedersen protocol* is almost as important as the Schnorr protocol and probably best known for its application within the Zerocoin protocol, an extension of the Bitcoin protocol that uses zero-knowledge proofs in order to further improve participant anonymity. The security of the Pedersen protocol is also based on the discrete logarithm problem yet slightly more complex than the Schnorr protocol, despite also only requiring three interactions. Again, a numerical example in figure 4 will support the basic theory.



Source. Own illustration.

Figure 4: Numerical example of the *Pedersen protocol*

This time, the security of the protocol is somewhat enhanced since the initial discrete logarithm problem is extended to the following form: $c = g^x * h^r \pmod{n}$. Initially, the individual variables are straightforward from the Schnorr protocol with one exception. A new variable h is derived from the public generator g and a random variable a in the following way: $h = g^a \pmod{n}$. Again, Paula wants to prove her knowledge of x without revealing any information:

(1) Paula randomly chooses the variables p and q. This enables her to compute $d = g^p * h^q \pmod{n}$ and send the variable d to Victor.

(2) Victor now randomly chooses a variable e and sends it to Paula.

(3) Paula then computes $u = p + e * x$ and $v = q + e * r$ and sends both variables u and v to Victor.

Victor can now compare Paula's outcome of $g^u * h^v \pmod{n}$ with his result of $d * c^e \pmod{n}$. As we have already seen before, if both receive

the same result, Paula has successfully proven her knowledge of x. The proof of the Pedersen protocol is similar to the proof of the Schnorr protocol. It can be seen in equation 2 which is included in appendix A for completeness. [Fra14] [Ped91]

4.1.3 zk-SNARKs

Interactive zk-proofs are very good to get the intuition of how zero-knowledge proofs actually work and what use cases they could possibly serve. However, in practice such back and forth interactions are not very useful, especially not in a public blockchain environment. Hence, a concept called *non-interactive* zero-knowledge proofs is much better suited to such an architecture. Non-interactive zero-knowledge proofs only require one single iteration to prove knowledge. Because these non-interactive proofs can later be verified by anyone, they can simply be published to everybody and basically be considered a digital signature.

One particular non-interactive proof construction is called *zk-SNARK*, which is an acronym for ***z**ero-**k**nowledge **S**uccinct **N**on-interactive **AR**gument of **K**nowledge* [BSCG+13]. The term *succinct* within the acronym refers to the fact that its proof length and verification time only depend on a security parameter and not on the size of the statement to be proven. Of course, this is very similar to a hash function, where for example the hash value of the SHA-256 algorithm will always have the same length (256 bits) according to the security parameter (256 bits). Consequently, one zk-SNARK proof has a length of only a few hundred bits and is verifiable within milliseconds. This allows one prover to convince many different verifiers simultaneously with only a single message.

Yuan [Yua19] provides a neat way of looking at a zk-SNARK as an analogy to a hash algorithm. Turning arbitrary computations into a zk-SNARK is comparable to turning arbitrary data into a hash value. And since verifying arbitrary computations is the core essence of Ethereum, zk-SNARKs are well suited to a public blockchain architecture. In such an environment, among many other applications, they could ensure that

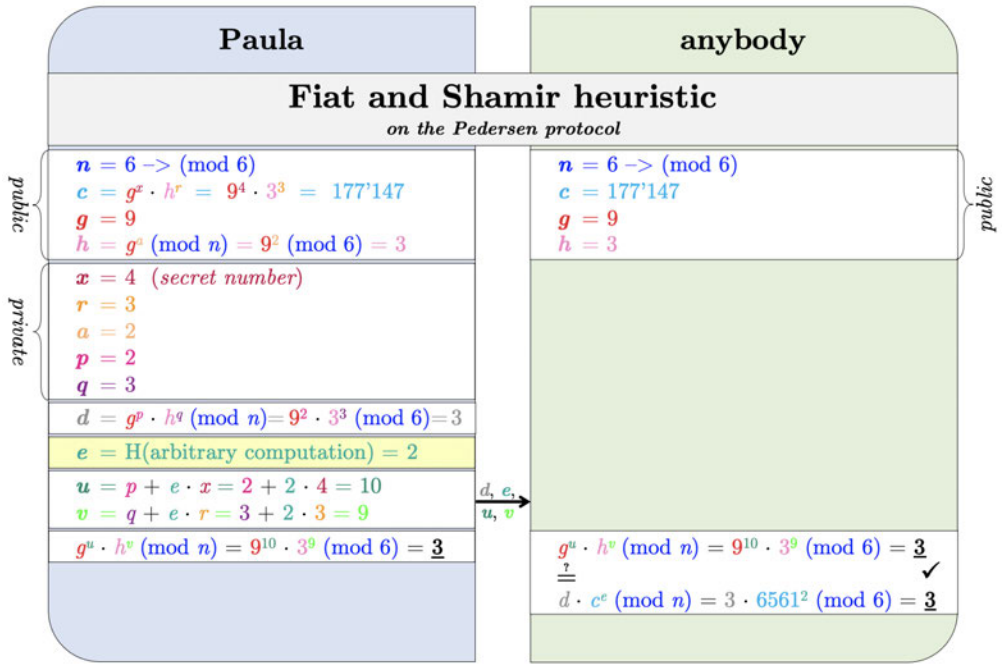
posting invalid or malicious transactions becomes impossible for any operator and thus guaranteeing the validity of a sidechain.

Despite the many bright possibilities that come with zero-knowledge proofs and zk-SNARKs, there is also a crucial drawback related to zk-SNARKs. The generation of such proofs is computationally intensive and therefore relatively time-consuming. This could pose a significant problem to a public blockchain like Ethereum that adds a new block to the blockchain every 15 seconds. Of course, the consequences of this fact will be addressed and further discussed in section 4.4. [BSCG+13] [Fra14] [Nit20] [Rei16] [Yua19] [Zca20]

4.1.4 Fiat and Shamir heuristic

Recall the two interactive zero-knowledge proofs of Schnorr and Pedersen from above. In 1986, Amos Fiat and Adi Shamir published a cryptographic approach that is known as the *Fiat and Shamir heuristic* [FS86]. This heuristic is a fairly simple approach to turning an *interactive* into a *non-interactive* zero-knowledge proof. The basic idea is to replace Victor's interaction, where he sends some random variable e to Paula, with a hash value of an arbitrary computation. This is based on the assumption that it is impossible for Paula to predict the outcome of this cryptographically secure hash function. Figure 5 illustrates a possible application of the Fiat and Shamir heuristic on the Pedersen protocol from above, successfully transforming it into a non-interactive zero-knowledge proof.

This intuitive example only intends to provide an idea on how interactive zero-knowledge proofs can be turned into non-interactive ones. Nevertheless, it is important to clarify that zk-SNARKs are not just based on a Schnorr or Pedersen protocol and turned into a non-interactive zero-knowledge proof by applying the Fiat and Shamir heuristic. In fact, a zk-SNARK is a proof system with a very complex construction that is based on several algorithms, which ultimately enables its succinctness. I refer the reader to [BSCG+13] for a detailed introduction to the construction of a zk-SNARK. [BSCG+13] [Fra14] [Kog19] [Nit20]



Source. Own illustration.

Figure 5: Possible application of the *Fiat and Shamir heuristic*

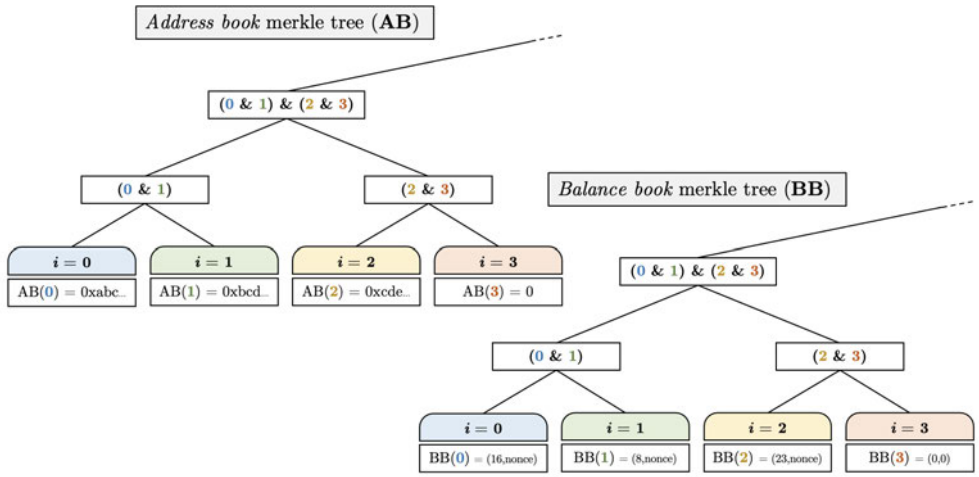
4.2 Construction

Now, let's return to zero-knowledge rollups and focus on their underlying architecture. Zk-rollups are managed by one single smart contract on the Ethereum mainchain, consisting of two 32 byte values as state variables. In zk-rollups there are two different types of actors. First, there are *transactors* who want to transfer tokens. Second, and somewhat similar to the operators in plasma, there are the *relayers* who collect and process a set of transactions off-chain. However, in zk-rollups they also generate zk-SNARKs that prove the validity of the whole transaction bundle. The zk-SNARKs are published on the Ethereum mainchain, along with the highly compressed transaction data via calldata[1], which is much cheaper than storing or computing the transaction data on-chain. In return, relayers are rewarded with a small transaction fee from every transactor.

[1]Calldata is a special read-only area that contains data parameters of an Ethereum transaction. [Sai19]

Traditionally, every ERC20 token transfer requires one Ethereum transaction, which costs a base fee of 21'000 gas. Yet, a zk-rollup smart contract bundles hundreds of transfers together and processes them into one single transaction with the help of a zk-SNARK. For that, the transaction data needs to be highly compressed in order to be published on-chain as efficiently as possible. Despite this highly compressed format, the smart contract is capable of dismantling all the individual transfers, with the zk-SNARK mass-validating each and every one.

This massive compression of transaction data is only possible due to a clever indexing trick, that involves the two 32 byte values within the smart contract. Each of these values represent a merkle tree, both able to store 2^{24} entries. The first tree is used as an *address book* (AB) and the second as a *balance book* (BB). Initially, both of these merkle trees are empty. Every branch in both merkle trees is indexed with an index number (i) starting from 0. The AB merkle tree stores an Ethereum address at the position (AB(i)) for a particular index i. The BB merkle tree stores the balance (along with a nonce) of this same address with the corresponding index i.

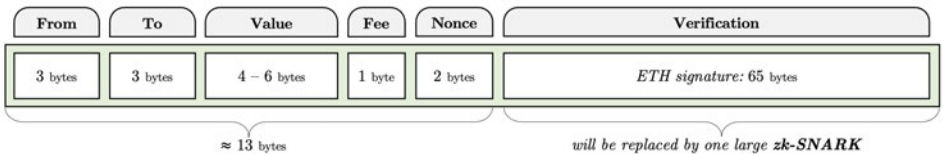


Source. Own illustration.

Figure 6: The two merkle trees within a rollup contract

A new transactor can register by showing that a merkle branch for a specific index number is equal to 0 (in AB) and thus empty, i.e. in figure 6, the branch AB(3) for the index number $i = 3$ is still empty. In addition to this, the previous position must not be empty, because the merkle tree has to be filled in order. Of course, this whole merkle tree is logged, so that the whole information within this tree is accessible to everybody and new transactors can register to the zk-rollup.

Using the corresponding index as a representative of actual Ethereum addresses is crucial, because it enables to shrink the size of the addresses from 20 bytes (*ETH address*) to only 3 bytes (*ETH index*). This applies to both, the sender's and the receiver's address. In theory, the whole Ethereum supply of around 110M Ether could roughly be represented using 6 bytes. Thus, a transaction *Value* of 6 bytes should be more than enough, while a *Value* up to only 4 bytes is more realistic in practice. The *Nonce* and the transaction *Fee* for the relayers are both very small. Hence, one single zk-rollup token transfer can be represented with only 13 bytes using the following format, illustrated in figure 7.

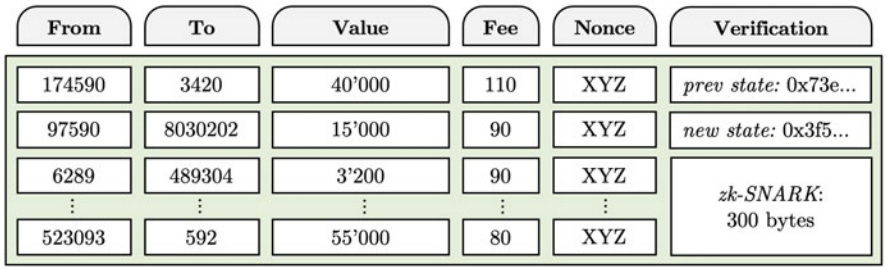


Source. Own illustration.

Figure 7: Zk-rollup transaction data

To submit a transaction, a transactor has to publish all the values *From*, *To*, *Value*, *Fee*, *Nonce* and his *Signature* as a verification to a relayer, as illustrated in figure 7. The relayer then gathers many transactions, bundles them into one single transaction and generates a zk-SNARK, which proves the validity of all signatures and balances. The main advantage of this rollup procedure is that it enables to omit all individual signatures which have a relatively large size of about 65 bytes. Figure 8 illustrates, how these signatures can easily be replaced by one single zk-SNARK

that is capable of verifying hundreds of individual signatures and has a size of a mere 300 bytes [BSCTV13]. Deploying one single zk-SNARK in an Ethereum transaction costs about 600'000 gas and the overhead, that covers basic transaction costs and additional log costs, uses additional 50'000 gas. In addition to bundling the transactions, generating a zk-SNARK and publishing everything to the mainchain, the relayer also updates the indexing merkle trees within the zk-rollup contract after every set of transactions. [But18] [But19b] [Wu19]



Source. Own illustration based on [But19b].

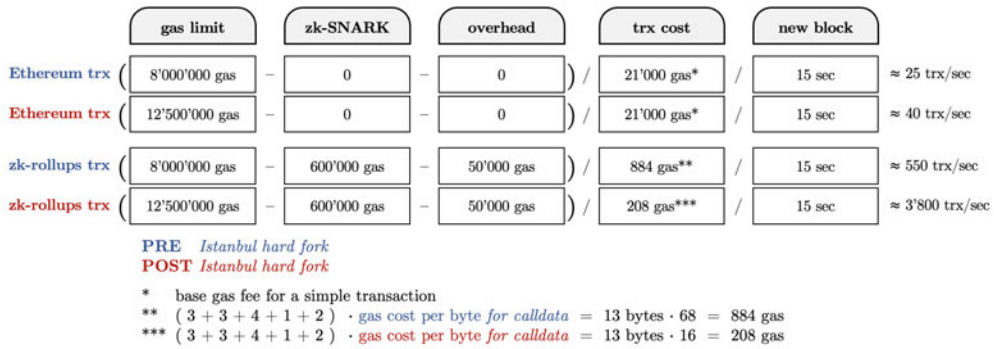
Figure 8: One zk-rollup transaction that contains a set of token transfers

4.3 Throughput

Back when Vitalik Buterin published an improved version of zk-rollups in September 2018, he used the following values to assume a potential transaction throughput. At that time, the cost of on-chain calldata was 68 gas per byte. This led to estimated costs of around 884 gas per zk-rollup transaction. The gas limit of one Ethereum block was at roughly 8'000'000 gas. This resulted in an approximate transactional throughput of about 550 transactions per second – which is by the way also the title of his post in 2018 [But18].

Yet, these calculations were made before the *Istanbul hard fork* took place at the end of 2019. Among many other improvements, this fork implemented the *EIP-2028*, which reduced the gas cost for calldata from 68 to only around 16 gas per byte [ASB+19]. Consequently, a transaction with

a size of around 13 bytes now costs only a mere 208 gas instead of 884 gas. In addition to this hard fork, Ethereum miners opted to increase the gas limit of an Ethereum block [The20]. This limit was initially increased from 8 million to 10 million gas at the end 2019 and further increased up to a limit of 12.5 million gas in mid 2020. Assuming the gas cost of a zk-SNARKs and of the overhead stayed roughly the same and taking the increased block limit and the reduced calldata costs into account, we now look at an approximate throughput of roughly 3800 transactions per second.



Source. Own illustration.

Figure 9: Throughput approximation of zk-rollups

Let's put this into a perspective: recall Ethereum currently averages around 25 transactions per second (with a potential ceiling of around 40 trx per second) and Visa records an average throughput of around 2000 transactions per second. Therefore, zk-rollups' current ceiling of almost 4000 transactions per second is a very decent throughput. Although these numbers are very high and appear promising, we should not overestimate them, since they are only theoretical throughput approximations. In practice, such large numbers of transactions volumes are currently not realistic. First, because there is currently no demand for such high throughputs and second, and more importantly, the actual implementation of zk-rollups still poses some challenges, that I am going to address in the following section. [But18] [But19b] [Del18] [SL19] [Wu19]

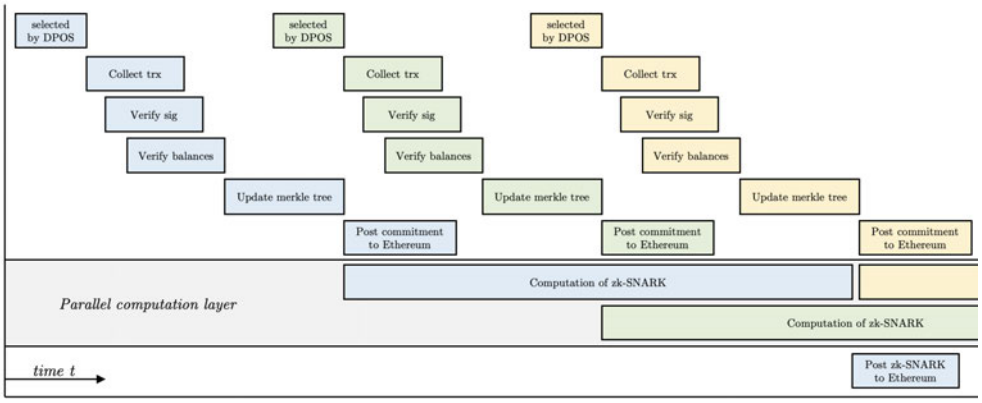
4.4 Implementation

As with every Ethereum scaling solution, there is not just one research group that is working on actual real-world implementations. In fact, this is the same for the concept of zk-rollups. There are many different researchers trying to develop a reasonable solution that can be implemented in practice. Therefore, I will focus on an approach by a research group called *Matter Labs* which is one of the largest research groups working on a real-world implementation of zk-rollups.

As previously mentioned, the generation of zk-SNARKs is the actual bottleneck of zk-rollups as it can take up to several minutes. Matching this to the Ethereum blockchain, where a new block is added every 15 seconds, turned out to be some challenge. Consequently, Matter Labs proposed a *commit-verify approach* that includes parallel computation of zk-SNARKs, but also introduces some inevitable latency into the system. In other words, one relayer submits a commitment of a transaction bundle on the Ethereum blockchain and starts to compute the zk-SNARK that verifies its validity. As soon as the corresponding zk-SNARK is successfully generated, the verification of the previous on-chain commitment will be published a few Ethereum blocks later. Of course, this comes at the cost of a transaction latency of a few minutes, until the computationally intensive zk-SNARK is generated and able to verify the validity. However, this commit-verify approach enables the introduction of a parallel computation layer, that increases the overall efficiency of zk-rollups. Since a relayer always starts with a commitment to a specific transaction bundle, the two merkle trees can be updated right away. As soon as the merkle trees are updated, the next relayer can start to commit to his next set of transactions and start computing his zk-SNARK simultaneously.

This proposed multiple-relayer model from Matter Labs uses a delegated proof-of-stake mechanism to randomly choose a new relayer, who then commits to the gathered and verified transactions and starts to compute the mass-validation – the zk-SNARK. This delegated proof-of-stake approach avoids a centralization of relayers.

Figure 10 provides an illustration, on how such a parallel computation model can be achieved. While parallel computation cannot decrease the fairly high latency of transactions, it certainly enables a significant increase of transaction throughput. [But18] [But19b] [GV19] [Wu19]



Source. Own illustration based on [GV19].

Figure 10: A multiple-relayer model for zk-rollups

4.5 Conclusion on zk-rollups

At first glance, zk-rollups provide a promising scaling solution for Ethereum. The general idea to use zero-knowledge proofs to tackle the data availability problem is brilliant. There is no liveness assumption involved. Exiting a zk-rollup contract can be achieved within minutes because it does not involve a tiring exit procedure. The potential scalability gains are enormous, even in worst case scenarios. Indeed, the implementation is not straightforward but definitely solvable. The introduced latency can technically be overlooked, since a transaction is almost secure as soon as the relayer posted his commitment to a set of transactions.

On the other hand, there are also some drawbacks related to the construction of zk-rollups. Currently, zk-SNARK constructions depend on an initial trusted setup between a prover and a verifier. This indicates that a set of public parameters has to be encoded within the protocol, in-

troducing some level of initial centralization, because they are coded by a small group of developers. Despite the fact, that this trusted setup is still essential today, there is a lot of research conducting this issue. One very promising approach aiming to create efficient and trustless zk-SNARKs was brought forward in late 2019 by Alexander Vlasov and Konstantin Panarin [VP19], both researchers at Matter Labs.

It is also worth mentioning that the scalability gains only refer to simple token transfers. A theoretical throughput of 3800 *transactions* per second technically only indicates 3800 *token transfers* per second. To this date, zk-rollups do not support commonly used smart contract standards. In simple terms, while zk-rollups can scale token transfers pretty efficiently, smart contract interactions still have to be performed on-chain as usual. However, according to Alex Gluchowski, co-founder of Matter Labs, there are approaches that may enable the support of turing-complete smart contracts on a zk-rollup chain in the near future. [Bin20] [Glu19b]

An arguably easier way to scale Ethereum smart contracts for general purposes might be to employ an alternative rollup solution called *optimistic rollups*. Therefore, I provide a profound introduction to optimistic rollups in the following section.

5 Optimistic rollups

Soon after Vitalik Buterin published the Ethereum whitepaper in 2013 [But13], the first layer 2 proposals had already surfaced in the Ethereum community. Retrospectively, these solutions already resembled the general idea of optimistic rollups quite closely. One of them was the concept of *shadow chains*, by Vitalik Buterin himself in 2014 [But14]. However, the real initial proposal for optimistic rollups was published by John Adler in mid 2019 [Adl19a], calling it *Minimal Viable Merged Consensus* back then. In fact, a few weeks later it was already conventionally called *optimistic rollups* by Karl Floersch [Flo19], because of its close similarity to zk-rollups.

First, I will introduce the concept of optimistic rollups and highlight some similarities and differences to zk-rollups. Thereafter, I am going to provide throughput approximations, before describing an important ingredient called *optimistic virtual machine*, that ultimately allows smart contract compatibility of optimistic rollups.

5.1 Concept

The concept of optimistic rollups is technically very similar to the concept of zk-rollups. Yet, there is one significant twist: Instead of using computationally intensive zk-SNARKs, optimistic rollups' validity relies on fraud proofs, as we have already seen with plasma before. Simply put, optimistic rollups are a combination of zk-rollups and plasma, combining the advantages of on-chain data availability with a very simple fraud proof mechanism. As a result, optimistic rollups claim to tackle the main drawbacks of zk-rollups: the very long generation process of a zk-SNARK and the lack of support for commonly used smart contract standards. However, in order to achieve full smart contract compatibility, optimistic rollups trade off some degree of scalability in comparison to zk-rollups.

Let's start with a simple intuition on how optimistic rollups work. Optimistic rollups are also based on a smart contract on the Ethereum blockchain and multiple aggregators act as nodes of the layer 2, similar to the relayers in the context of zk-rollups. Every aggregator has to deposit a bond when posting a new rollup block, which prevents them from misbehaving. Let Tommy be an *honest transactor* and Alfie be a *bonded aggregator*.

Tommy sends a transaction containing some solidity contract code to Alfie, who verifies this transaction and immediately returns a receipt back to Tommy. This receipt is a guarantee for Tommy that his transaction will be processed and included in a rollup block. Alfie processes Tommy's transaction locally, among many other transactions, and computes the new state root of the optimistic rollup block. Then, he sends the rollup block (the new state root, to be precise) to the Ethereum smart contract, along with a bond. The smart contract only records the hashes of the rollup blocks and does not interpret or execute anything. The whole transaction data and state root are only made available to everybody through calldata, as we have previously seen with zk-rollups. Up until now, the procedures are more or less straightforward from zk-rollups, apart from the fact that Alfie would now start to generate and later publish a zk-SNARK to the Ethereum blockchain.

In optimistic rollups, however, Tommy's transaction is secured by fraud proofs. If anyone else, let's say Polly, should discover that Alfie posted an invalid state root, she can submit a fraud proof by providing evidence of its invalidity. Then, the smart contract on layer 1 (or ultimately the EVM) checks weather Polly's claim was valid or not. If her claim was true and the block was proven invalid, this would slash Alfie's (and every following) block and his deposited bond. Part of the bond will be rewarded to Polly and part of it will be burned by the EVM. Tommy's initial transaction would then be reconsidered by some other aggregator.

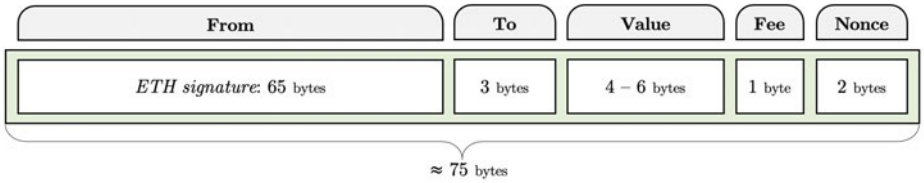
However, one major problem is that this fraud proof mechanism introduces a relatively long withdrawal period, in order to give the community enough time to challenge malicious blocks. Only after this challenging

period of about one week has elapsed, the transaction becomes final and Alfie can withdraw his deposited bond. A more thorough introduction on the consequences of a long transaction latency will follow in section 6.3.1. [Adl20] [But19b] [Flo19]

5.2 Throughput

Optimistic rollups use basically the same technique as zk-rollups to provide on-chain data-availability. The aggregators publish the transaction data and the state root onto the Ethereum blockchain through calldata. Again, the Ethereum blockchain itself is solely used as the data-availability layer. One might argue that all the layer 1 nodes still have to process all data coming from the rollup chain. However, since Ethereum nodes do not have to apply any logic to on-chain calldata, it is much cheaper in gas costs than actual on-chain storage and therefore much more efficient. Furthermore, recall that the *EIP-2028* introduced by the *Istanbul hard fork* in the end of 2019 reduced the gas costs of calldata by another four times.

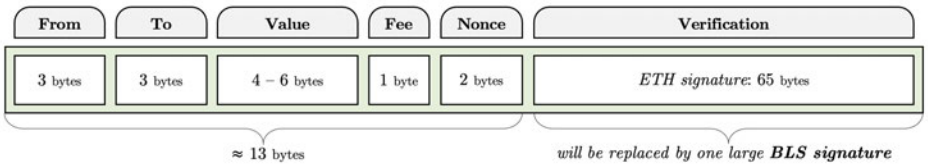
Again, let's us look at simple token transfers, in order to be able to compare the throughput ceiling of optimistic rollups and zk-rollups. Yet, bear in mind, that optimistic rollups, unlike zk-rollups, are not limited to token transfers only. Recall zk-rollups' indexing from section 4.2, which enables a large compression of Ethereum addresses. Similarly, the optimistic rollup contract on the Ethereum blockchain also employs these two 32 byte merkle trees to compress the Ethereum addresses. However, because optimistic rollups do not use zk-SNARKs that are able to replace all individual signatures, the sender's signatures have to be included in every transaction. Having to include such a large signature with a size of 65 bytes increases the size of the whole transaction by a factor of five, from about 13 bytes to around 75 bytes – this includes the *Sender's signature*, *To*, *Value*, *Fee* and a *Nonce*, as illustrated in figure 11.



Source. Own illustration.

Figure 11: Optimistic rollup transaction data

A larger amount of data per transaction obviously results in higher gas costs. Therefore, optimistic rollup aggregators are forced to pack less token transfers into one Ethereum transaction until the gas limit of a block has been reached. Thus, before the EIP-2028 was introduced in the Istanbul hard fork, optimistic rollups could only reach a throughput of around 100 transactions per second. Today, after the gas cost reduction of calldata and the gas limit increases of Ethereum blocks have been introduced, optimistic rollups can scale up to around 700 transactions per second. Again, comparing this to a current throughput of around 25 transactions per second, optimistic rollups can offer a pretty decent throughput increase.

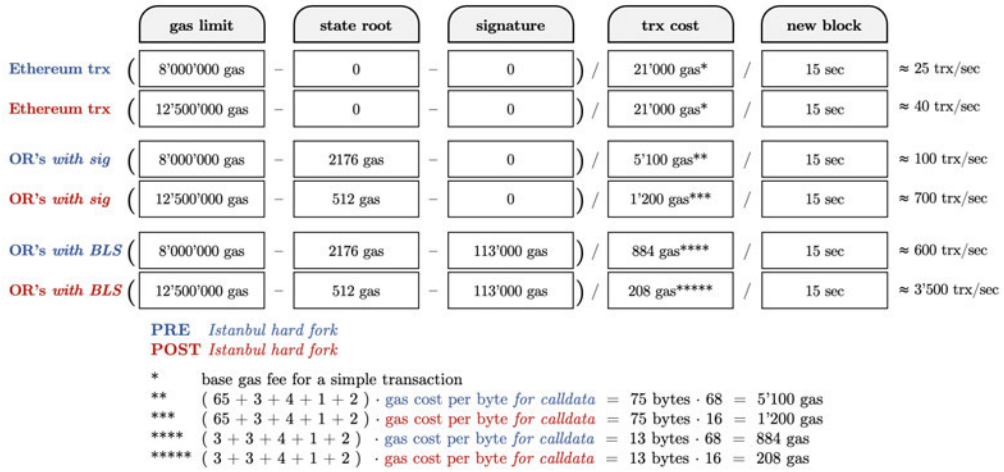


Source. Own illustration.

Figure 12: Optimistic rollup transaction data *using BLS signatures*

In a recent reddit post from August 2020, Vitalik Buterin pointed out that optimistic rollups could technically switch to *aggregate BLS signatures* pretty easily, which would replace the large 65 byte ECDSA signature within an optimistic rollup transaction. Without going into too many details, BLS signatures could work in a very similar way as the zk-SNARKs work in zk-rollups. They would replace all individual signatures with only one large BLS signature, that verifies the correctness

of all individual transactions. This would also introduce some fixed gas costs of around 113'000 that cover for the large BLS signature. On the other hand, it would also allow us to shrink the transaction size back to only about 13 bytes. Hence, a switch to some form of BLS signatures could eventually scale optimistic rollups up to zk-rollup's territory in terms of raw throughput numbers, that reach beyond 3000 transactions per second.



Source. Own illustration.

Figure 13: Throughput approximation of optimistic rollups

Note that these throughput numbers are only estimated values and are neither perfectly accurate nor realistic in the next couple of years. These theoretical approximations are solely based on the data availability bottleneck introduced by the Ethereum block gas limit. [Flo19] [Fos20] [Wu19]

I used a python script for a more accurate throughput approximation of these different optimistic rollup scenarios. I included this script in appendix B and in this GitHub repository. This script clearly draws some inspiration from Karl Floersch's work on the same topic.

5.3 Optimistic Virtual Machine

Now, let's briefly focus on a vital ingredient of optimistic rollups, the *Optimistic Virtual Machine*, or short the *OVM*. In mid 2019, the idea for an OVM was introduced [Eth19] by a research group then called *Plasma Group*. They have always been at the forefront of plasma research since 2017. However, in early 2020, they changed their name into *Ethereum Optimism*, to emphasize their extensive research in the direction of optimistic rollups – the spiritual successor to plasma. This underlined a general shift within the Ethereum Community from the once very promising plasma to state-of-the-art optimistic rollups.

Recall, the *EVM*, short for *Ethereum Virtual Machine*, is an environment that handles the deployment and execution of Ethereum smart contracts. Every transaction that involves state updates is processed and executed by the EVM. It is a virtual environment that is distributed across all nodes of the network and therefore also referred to as the *world computer*. The OVM is the exact counterpart of the EVM but implemented on layer 2. The development of the OVM has been one of the most important innovations for optimistic rollups, because it allows an implementation of arbitrary smart contract logic natively on layer 2. In simple terms, the OVM allows everyone to move a solidity smart contract onto the much cheaper and faster rollup chain. In addition to this, the OVM could also act as an arbitration court that checks the validity of fraud proofs. As a result, fraud proofs become much more efficient.

The basic setup of the OVM contains three integral components: A *transpiler algorithm* translates ordinary EVM bytecode into an OVM-compatible format, in order to run any Ethereum smart contract off-chain. A *safety checker algorithm* verifies that the translation of the transpiler worked seamlessly and no exceptions occurred. An *execution manager* simply stores and executes all OP codes, or simply put, processes and executes the whole smart contract. [Eth19] [Eth20a] [Eth20b] [Flo20] [WZS19]

5.4 Smart contract compatibility

Let's examine why optimistic rollups are well suited to handling turing-complete smart contracts by focusing on three essential properties. In order to natively support and handle complex smart contracts, a state machine has to fulfill the following three properties: The head state has to be *valid*, *live* at all times and *available* to every party.

Note that the term *head state* refers to a giant data structure, which connects all individual states and transactions of all accounts, hashes them together and stores them into one single state root – the head state. It represents the current, most recent canonical state of the blockchain, sidechain or rollup chain. For example, the Ethereum blockchain has one (and only one) current head state, which is one single point of truth. Of course, after a new block is created and added to the chain, the head state changes. Technically, and if the three properties are met, every participant could recompute and therefore verify the actual head state because it is based on all past transactions. Now, let's look at how optimistic rollups can satisfy all of these three properties.

5.4.1 Validity

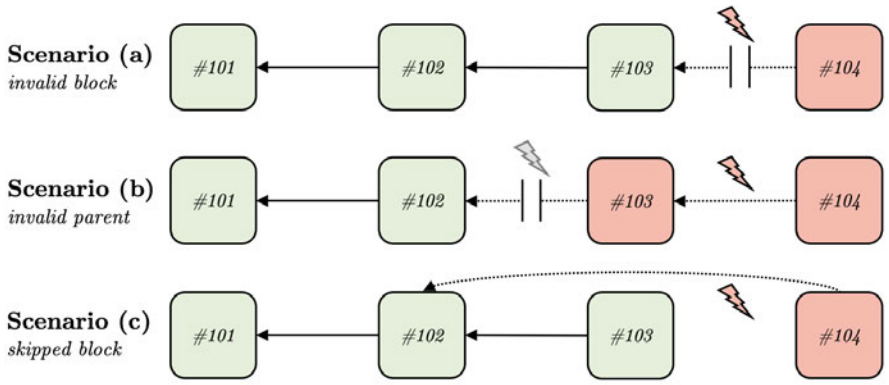
Zk-rollups use complicated zk-SNARKs to ensure validity which are indeed brilliant for single token transfers but not yet efficient enough to cope with turing-complete smart contracts. In contrast, optimistic rollups are based on a cryptoeconomic validity game, that does not involve any extensive computations and is thus very efficient. In optimistic rollups, any user can become an aggregator by depositing a security bond. An aggregator collects the incoming transactions, computes the new head state and commits the new block to the rollup contract on the Ethereum blockchain.

In general, there are the following three scenarios that lead to invalid optimistic rollup blocks and thus to an invalid head state:

(a) A block does not contain the correct transactions or an error occurs.

(b) A block is based on a previously invalid block.

(c) A block skips a valid block and is not part of the longest chain.



Source. Own illustration based on [Flo19].

Figure 14: Three scenarios that lead to invalid optimistic rollup blocks

Optimistic rollups are permissionless, meaning anyone can become a validator, and trustless, indicating a transactor does not have to rely on only one aggregator. All transactors and aggregators within the network are incentivized to verify the validity of every block, since proving a block invalid wins them a slice of the malicious aggregator's deposit. Yet, it is not very realistic to expect normal users to monitor and check all transactions, because if they had such technical capabilities, they would probably become an aggregator themselves.

However, because of *scenario* (*b*) which states that every block based on a previously invalid block is also invalid, all aggregators are naturally incentivized to check the validity of the previous blocks, before publishing their own. Hence, as long as there is at least one honest aggregator (or any other user checking the validity) within the network, the transactions are secure and only one valid head state exists.

5.4.2 Liveness

The liveness of the head state is also referred to as the *censorship resistance*, which means that malicious aggregators cannot censor or ignore a single transaction. Referring to the example from before, no aggregator will ever publish his new block on top of an invalid block. Therefore, every aggregator will (at least) verify the validity of the previous block, before publishing his own. Should a previous block be invalid, an aggregator would file a fraud proof, slashing the invalid block, and recompute the head state based on the last valid block in the rollup chain.

Note that optimistic rollups use a non-interactive verification game, indicating that fraud proofs do not need interactions and can be proven immediately. Hence, the rollup chain will not come to a halt, even if it contains an invalid block at one point, because an honest aggregator will always submit a block, which forks around an invalid block. Therefore, the liveness assumption of optimistic rollups is given. To summarize, the head state of the optimistic rollup chain is always live, as long as there is at least one non-censoring aggregator.

5.4.3 Availability

As previously described, optimistic rollups achieve on-chain data availability in a similar way to zk-rollups. Simply put, calldata is used as an availability oracle for transaction data and the current head state. Hence, all actions and the most recent information about the optimistic rollup chain are always available to everybody on layer 1. Since there are no restrictions to download the current head state, this property is also satisfied.

Notice that the Ethereum blockchain itself is a state machine that satisfies all of these three properties, of course. In addition to this, Ethereum has an EVM that processes all the state changes. Currently, optimistic rollups are the only layer 2 solution that supports turing-complete smart

contracts because they have their very own OVM and they satisfy all three head state properties with relative ease.

To emphasize the uniqueness of optimistic rollups satisfying all three properties, we can compare them to other popular scaling solutions. For example, state channels cannot provide the head state availability to everybody, since they are not open and rely on a closed set of participants. Plasma has some sudden liveness failures and, as a consequence, cannot always ensure the liveness of the current head state. In addition to this, the transaction data of a plasma chain is only available to sidechain participants and not available on the Ethereum blockchain. Of course, another interesting candidate are zk-rollups, because they would technically also satisfy all the three properties from above. Nevertheless, zero-knowledge proofs are not yet advanced enough to fully support an EVM (or in this case probably a construct called *zkVM*), even though current research suggests potential future implementations of such. [Adl19b] [But19b] [Flo19] [Flo20]

5.5 Conclusion on optimistic rollups

Optimistic rollups are touted as the one off-chain solution that most closely resembles actual on-chain scaling and thus are sometimes referred to as the *layer 1 of layer 2 solutions*. Optimistic rollups are trustless as well as permissionless. They excel at providing full support of currently known Ethereum smart contracts. However, the use of fraud proofs instead of zk-SNARKs trades in some degree of pure payment throughput. A possible implementation of BLS signatures in the future could make the scalability deficit compared to other layer 2 solutions become even less significant.

Because transactors receive almost instant receipts for every transaction they have sent to an aggregator, the long timeout periods before rollup transactions are finalized and locked are almost negligible in practice. Yet, I will still provide a further assessment of the consequences caused by these long finality times in section 6.3.1. The chances of stumbling

across a malicious aggregator are very slim and even if such an event may occur every now and then, almost instant penalizations prevent the rollup chain and all affected transactors from further damage. The overall usability of optimistic rollups is good and very intuitive for Ethereum users, and the use cases are more or less straightforward from layer 1, as most smart contracts can be easily migrated to optimistic rollups.

6 Comparative analysis

Despite the fact that some comparisons between optimistic and zero-knowledge rollups have already been drawn throughout the whole thesis, this section will dive deep into further and more specific comparisons between the two rollup flavors.

6.1 Scalability

In fact, the throughput numbers of both rollup constructions are only approximations on the basis of the data availability bottleneck. This refers to an approach that solely considers how many transactions could possibly be put into a single Ethereum block before its gas limit is reached. Thus, both solutions are upper-bounded by their data-availability mechanism. This also explains why a solution like plasma, which completely abandons on-chain data availability, can scale significantly higher than rollups. The limiting factor of rollup transactions are Ethereum's block gas limits. Another crucial ingredient in this throughput approximation are the actual gas costs per byte of calldata, which is used to publish the transaction data onto the Ethereum blockchain.

Over the course of the past 15 months, there have been several implementations that benefitted both rollup solutions drastically. Firstly, the gas limit per block has been increased from 8 million to about 12.5 million gas and secondly, the Istanbul hard fork introduced the EIP-2028 that has cut the gas cost for calldata in quarters. While this improved rollups' potential throughout ceilings in comparison to other solutions, this was the same for both rollup constructions and did not favor one or the other.

Hence, the decisive ingredient between the two rollup solutions is still the size of the transaction data that will be published via calldata. By using zk-SNARKs all individual signatures become obsolete, which allows zero-knowledge rollups to shrink the size of one transaction down to a mere 13 bytes of calldata. In contrast, optimistic rollups, where

the signature of every transactor has to be included, can compress the transaction size only to 75 bytes of calldata, about five times the size of a zk-rollup transaction. Of course, this ratio is congruent with the ratio of zk-rollups' raw throughput ceiling of about 3800 token transfers per second compared to optimistic rollups' 700 transfers (without a future implementation of BLS signatures). Despite the fact that both values are a significant increase compared to Ethereum's throughput today, there are still a few points that need to be considered.

Of course, optimistic rollups trade in some degree of pure token transfer speed in order to allow arbitrary smart contract logic off-chain. Since smart contract interactions are not as straightforward as simple token transfers, a potential throughput number of about 700 *transfers* cannot be changed to 700 *smart contract interactions* per second. Optimistic rollups are definitely going to scale the amount of smart contract interactions, but not up to such high ceilings. Realistically, the throughput of optimistic rollups could even be capped at a few hundred transactions per seconds. This could prove to be crucial in order to remain compatible with Ethereum's current EVM on layer 1, which would otherwise not be able to keep up with such high throughputs.

Ultimately, let's not forget that Ethereum's main differentiator, compared to other public blockchains, such as Bitcoin for example, are its turing-complete smart contracts. Since optimistic rollups are the only layer 2 solution that allows to us scale smart contracts, it should not be underestimated by only looking at transactional throughput ceilings. In addition to this, zk-rollups' theoretical throughput of several thousand transfers per second have neither been needed nor tested in practice, hence should also not be overestimated. Both rollup solutions can deliver significant improvements to Ethereum when it comes to scalability gains and most importantly, they are the only layer 2 solutions that can tackle the data availability problem. [Adl20] [ASB+19] [Glu19b] [The20]

6.2 Security

An essential aspect of any scaling solution is its security. Especially because the transaction data is published onto the Ethereum blockchain, rollups are already very secure in general. Yet, there are still certain properties where the two rollup approaches differ, which might affect their respective security levels. In particular, I focus on the following two incremental security aspects when comparing optimistic and zero-knowledge rollups: *decentralization* and *manipulation resistance*. For the latter condition, I will briefly introduce and discuss one specific security concern only affecting optimistic rollups.

6.2.1 Decentralization

Both rollup solutions are implemented through a single smart contract containing all the funds from the rollup chain. As rollups are set to become the most popular layer 2 solution in the future, the volumes that will be held by a single smart contract are going to be massive. Thus, it is pretty obvious that these rollup contracts will probably be targeted by many hackers from all around the world. Because such high profits are at stake, sooner or later all possibilities for tiny loopholes will be exploited, no matter how expensive or complex they may seem.

Of course, such massive concentration risks are not quite on par with common decentralization principles, that a trustless public blockchain typically incorporates. On top of that, optimistic rollups rely on an assumption that at least 1-of-N aggregators has to be honest. Especially in the beginning, when there will not be an incredibly large number of aggregators, this could potentially lead to some sort of collusion between a small set of malicious aggregators. Although such attacks are very hard to execute, they are also not infeasible and definitely more likely with only a few aggregators compared to several thousand nodes on the Ethereum blockchain.

Considering an early implementation of optimistic rollups with only a few users and aggregators, there are definitely more centralization risks attached to optimistic rollups compared to zk-rollups. Ultimately, both rollup solutions are implemented similarly with only one single smart contract on the Ethereum blockchain and thus both approaches sacrifice some sort of decentralization.

6.2.2 Manipulation resistance

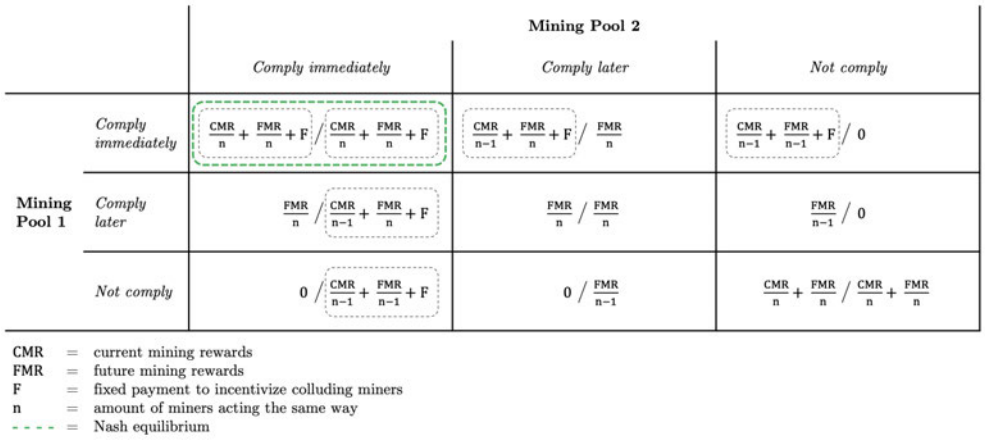
First of all, zero-knowledge rollups are completely immune to any manipulation or censorship attack, since the included zk-SNARK is a compressed representation of every individual signature. There is simply no way for any aggregator to steal funds or corrupt the current rollup state of a zero-knowledge rollup contract.

On the other hand, optimistic rollups use fraud proofs instead of zero-knowledge proofs. Their security depends on the assumption of a strong censorship resistance of the underlying Ethereum blockchain. Based on game-theoretic principles, the layer 1 can theoretically provide such an ordinary censorship resistance for on-chain Ethereum transactions. Nevertheless, Alex Gluchowski describes a scenario [Glu19a], where a bribing system could be created by using mechanism design, which would ultimately break the censorship resistance of the underlying Ethereum mainnet. Especially in the context of a large honeypot, say a single rollup contract containing more than $100M worth of funds, such anti-mechanisms could be exploited. First, I will introduce a scenario that gives an intuition on how such an attack could possibly work and later discuss its feasibility.

A malicious aggregator rents an enormous amount of GPUs to obtain 51% of Ethereum's hashing power, which would cost him several hundred thousand dollars per hour [Dic21]. Then, the attacker posts a malicious rollup transaction that transfers all funds from the optimistic rollup to his own account and immediately censors all incoming fraud proofs. This is possible because he has the majority of hashing power within the net-

work and can therefore easily start a soft fork that grows faster than the existing chain. On a side note, this also earns him all the mining rewards in the process, which hypothetically lowers his costs of the attack significantly. Thereafter, he publicly announces his ownership of the funds from the rollup contract and incentivizes every miner, who starts accepting his soft fork, with slices of his scammed assets. Consequently, he can slowly start to decrease his hashrate again and as soon as the challenging period of the optimistic rollup transaction has expired, the remaining funds are successfully secured.

Alex Gluchowski claims that this is almost a zero-cost attack for the attacker and every rational and rent-seeking miner is going to comply with his soft fork. Since the attacker only needs half of the hashrate within the network opting to comply with his soft fork, the miners will be confronted with conflicting interests, because they know that other miners also benefit from complying. In fact, at the time when the soft fork first occurs and the miners are offered a slice of the stolen funds, immediately supporting the soft fork of the attacker is the best response for every miner – a *Nash equilibrium*[2].



Source. Own illustration.

Figure 15: Game-theoretic model of an attack on optimistic rollups

[2]A Nash equilibrium is the optimal outcome of a game, where no player has an incentive to deviate from his best answer, given the strategy choices of the other players [Nas50].

Consider the following underlying assumptions for the game-theoretic model in figure 15. Only if there are not enough miners complying with this soft fork (two complying mining pools are sufficient in our model), the attacker will resign, and the attack will have failed. As a result, the honest chain would then become the longest chain again. If this was the case and the attack has failed, it would have been beneficial for both mining pools not to have complied with the attacker. However, assuming that miners act rational and are immune to the social costs of a betrayed Ethereum community, this scenario will never even occur, since the option *comply immediately* is a Nash equilibrium. The figure 15 states the rewards for two mining pools during this attack under the assumptions from above. It clearly shows why complying immediately is a Nash equilibrium. Bear in mind, the value 0 indicates no rewards. Hence, given a miner has still an energy consumption, this rather implies a loss.

In fact, with the current proof-of-work consensus mechanism, there is no method of punishing misbehaving miners. However, with a future introduction of a proof-of-stake algorithm, the community could theoretically punish such controversial miners by slashing their stake. Yet, it is still highly debatable if a consensus could be reached to unilaterally punish some individual miners. It is even more unrealistic when considering that the majority of hashing power within the network would be the subject of punishment. Therefore, the long-awaited switch to proof-of-stake will also fail to prevent such a fraud proof censorship attack.

A direct consequence, to minimize the chances of such an attack, was to increase the challenging time and thus the finality time of optimistic rollups. Setting the challenging time to at least one week would provide more time for the community to coordinate on a counter procedure. Additionally, upholding a large hashrate for an extended period of time, complicates an attack that did not succeed instantly. Furthermore, I would raise some doubts on how a single attacker could possibly obtain and preserve a large enough hashrate to initiate such an attack on Ethereum in the first place. Yet, such an attack on optimistic rollups

is definitely not impossible, especially when considering that such an attack ultimately comes at relatively low costs compared to a possibly large honeypot like an optimistic rollup contract.

Ultimately, smaller optimistic rollup implementations with a challenging period of more than one week should technically be fine for the moment. The general risk of a successful censorship attack on Ethereum is not very large, but definitely not negligible at this point in time. Nevertheless, bear in mind that the larger the funds within a rollup contract, the higher the chances of a successful attack, because an attacker has more financial arguments to bribe miners. Considering that zero-knowledge rollups are inherently immune to any kind of the above-described censorship attacks, they are definitely the better rollup solution in terms of security, especially when it comes to manipulation resistance. [But19a] [Dai19] [Glu19a] [Glu19b] [LK20]

6.3 Usability

Since both rollup solutions scale pretty decently, other parameters should also be considered instead of only looking at security and raw transactional throughput. Therefore, the general usability may prove to be the decisive aspect to choose between the two rollup approaches. First, let us compare the *latency* of the two systems, before focusing on how fast funds can be *withdrawn* from the rollup contract.

6.3.1 Latency

The latency of a rollup system can also be described as the time until a transaction is verifiably finalized. The different latency times between both rollup approaches are again due to the nature of optimistic rollups' fraud proofs and zero-knowledge rollups' validity proofs. For the latter rollup solution, transaction finality is pretty straightforward. As we have seen in section 4.4, zk-rollups are probably going to be implemented via a *commit-verify approach*. The commitment of the transaction happens

almost instantly, whereas the actual verification will be published a few Ethereum blocks later and depends solely on the generation time of zk-SNARKs.

Currently, the computation of a zk-SNARK typically takes about 20 minutes. Hence, zk-rollup transactions can only be considered as final after this period, when the verification of the initial commitment can be published on the Ethereum blockchain. However, as the need for faster zk-SNARK computation increases, specialized hardware is naturally going to improve, allowing for a significant reduction in generation times. In addition to this, switching from CPU-based to GPU-based approaches may further accelerate this process, as we have previously experienced with the improvement of hash computations. Thus, it is reasonable to expect that the latency of zk-rollups will decrease from about 20 minutes to only a few minutes in the not too distant future.

On the other hand, when assessing the latency of optimistic rollups, different levels of transaction finality have to be considered: *full finality*, *subjective finality* and *bonded finality*.

Because of the afore-mentioned slight security concerns in the context of optimistic rollups, the challenging period for fraud proofs has to be at least one week. Hence, the finality time of optimistic rollups is fixed at more than one week and should not be lowered to assure a certain security level. This is what I refer to as *full finality*, since any transaction could theoretically be reverted within this challenging period.

A different finality approach is called *subjective finality*. This refers to the fact that any valid transaction that is posted to the Ethereum blockchain can technically be considered final, since only invalid rollup blocks can be rolled back. Remember, optimistic rollups do not rely on extensive computation of validity proofs and as a result, transactions can be posted on-chain almost immediately. Therefore, optimistic rollups have instant subjective finality. This is only true to a certain extent, however, if the rollup block is based on a pervious block that was deemed to be invalid, it will also be rolled back. Yet, there is a way to check whether this is the

case or not. Any transactor or aggregator can go back the entire challenge period and start verifying every transaction and the corresponding state root. If every previous block up until the transactors' transaction is valid, it is basically impossible that their own transaction will be rolled back. However, validating every transaction of the entire challenge period is time-consuming and can become quite expensive, because you have to download the whole rollup state and execute every single transaction.

The third and most vague approach to finality is only based on the game-theoretic incentives of an optimistic rollup aggregator. I call this the *bonded finality*. In theory, no aggregator has any incentives to post invalid blocks or base their blocks on any other previously invalid block, because they risk losing their deposited bond. Recall, that a transactor receives an instant receipt from the aggregator, stating that he will include this transaction in the next rollup block. As a matter of fact, one could argue that this implies instant transaction finality in optimistic rollups, as long as an aggregator's bond at stake is large enough. However, optimistic rollup blocks are theoretically only final after the challenge period of one week has elapsed and any claim of a faster finality time than the actual challenging period will always be coupled with some security compromises. [Adl20] [Bel19] [Glu19b] [Glu20]

6.3.2 Withdrawal times

In terms of general usability, fast withdrawals from layer 2 back to layer 1 are pretty much a necessity. Yet, the situation is pretty similar to the finality times: Withdrawing funds from a zero-knowledge rollup contract back to the Ethereum blockchain takes about 20 minutes and could be decreased significantly in the coming months. On the other hand, optimistic rollup transactions have to reach full finality until a user can exit the rollup chain. Therefore, it takes more than one week until funds can be withdrawn from the optimistic rollup contract. Again, zero-knowledge rollups have a significant advantage over its optimistic counterpart when it comes to fast withdrawals.

Nevertheless, this fundamental deficit could be overcome with the help of liquidity providers. They can provide access to funds on the Ethereum mainnet, while the actual funds of a transactor are still locked in a rollup contract. Especially for optimistic rollups, this concept has great potential to improve the usability significantly. Yet, zk-rollup could also benefit from such instant withdrawals for some specific use cases. Bear in mind, however, that these liquidity providers will charge a risk premium and zk-rollups' 500 times faster withdrawal times, in comparison to optimistic rollups, will definitely be reflected in their prices. Thus, zero-knowledge rollups also prevail in terms of liquidity pricing. Ultimately, liquidity providers will bring instant withdrawals to both rollup approaches and drastically enhance their user experience. [Glu20] [LK19] [LK20]

7 Conclusion

In this thesis, I introduced and compared two very promising solutions to scale Ethereum, both of which enable higher throughput of Ethereum transactions without compromising either decentralization or security too severely. One major advantage of layer 2 solutions is that several different scaling approaches can coexist and serve different use cases. Therefore, despite the fact that both solutions have some advantages in comparison to one another, we ultimately do not have to choose one or the other. In fact, I would even take this to the extreme and argue that the two solutions can complement each other perfectly. Zero-knowledge rollups are reasonably close to a real-world implementation, that allows for imminent scalability gains of simple token transfers. On the other hand, due to their ability to scale smart contracts, optimistic rollups will give zk-rollups the time to eventually support EVM-compatible smart contracts as well in the near future.

Furthermore, we should not dismiss the possibility that there might be yet another twist in the hunt for a realistic and secure Ethereum scaling solution anytime soon. Nevertheless, after many years of extensive research for scaling solutions, it is now encouraging to see that a large part of the Ethereum community seems to have finally gathered around rollups as their most sensible scaling solution for Ethereum.

The upcoming months will be crucial for Ethereum and it will definitely be exciting to see the continuing transition to Ethereum 2.0 in combination with the first rollup implementations going live. Whichever solution ultimately prevails and emerges as the scaling solution of choice, there will only ever be one real winner: the Ethereum community.

References

[ABSB18] Mustafa Al-Bassam, Alberto Sonnino, and Vitalik Buterin. Fraud and Data Availability Proofs: Maximising Light Client Security and Scaling Blockchains with Dishonest Majorities, September 2018.

[Adl19a] John Adler. Minimal Viable Merged Consensus, June 2019.

[Adl19b] John Adler. The "Why"s of Optimistic Rollup, November 2019.

[Adl20] John Adler. Optimistic Rollups for the Rest of Us, November 2020.

[AQC19] John Adler and Mikerah Quintyne-Collins. Building Scalable Decentralized Payment Systems, April 2019.

[ASB+19] Alexey Akhunov, Eli Ben Sasson, Tom Brand, Louis Guthmann, and Avihu Levy. EIP-2028: Transaction data gas cost reduction, May 2019.

[BBK+13] Endre Bangerter, Stefania Barzan, Stephan Krenn, Ahmad-Reza Sadeghi, Thomas Schneider, and Joe-Kai Tsay. Bringing Zero-Knowledge Proofs of Knowledge to Practice. In *Security Protocols XVII*, pages 51–62. Springer Berlin Heidelberg, 2013.

[Bel19] Alexandre Belling. Reducing the verification cost of a SNARK through hierarchical aggregation, March 2019.

[Bin20] BinanceAcademy. zk-SNARKs and zk-STARKs Explained, 2020.

[BSCG+13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. SNARKs for C: Verifying Program Executions Succinctly and in Zero Knowledge. In *Advances in Cryptology – CRYPTO 2013*, pages 90–108. Springer Berlin Heidelberg, 2013.

[BSCTV13] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture. Cryptology ePrint Archive, Report 2013/879, February 2013.

[But13] Vitalik Buterin. A Next Generation Smart Contract and Decentralized Application Platform, November 2013.

[But14] Vitalik Buterin. Scalability, Part 1: Building on Top, September 2014.

[But18] Vitalik Buterin. On-chain scaling to potentially 500 tx/sec through mass tx validation, September 2018.

[But19a] Vitalik Buterin. Responding to 51percent attacks in Casper FFG, October 2019.

[But19b] Vitalik Buterin. The Dawn of Hybrid Layer 2 Protocols, August 2019.

[CDE+16] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, Dawn Song, and Roger Wattenhofer. On Scaling Decentralized Blockchains. In *Financial Cryptography and Data Security*, pages 106–125. Springer Berlin Heidelberg, 2016.

[Dai19] Philip Daian. DevCon V talk: Designing Smart Contracts With Free Will, October 2019.

[Del18] Jaime Delgado. On-chain scaling with full data availability. Moving verification of transactions off-chain?, October 2018.

[Dic21] Tom Dickman. PoW 51percent Attack Cost, January 2021.

[Eth19] EthereumOptimism. Introducing the OVM, June 2019.

[Eth20a] EthereumOptimism. Optimistic Virtual Machine Alpha, February 2020.

[Eth20b] EthereumOptimism. OVM Deep Dive, May 2020.

[Flo19] Karl Floersch. Ethereum Smart Contracts in L2: Optimistic Rollup, August 2019.

[Flo20] Karl Floersch. EthCC 2020 talk: Optimistic Virtual Machine: Full Ethereum Smart Contracts with Optimistic Rollup, March 2020.

[Fos20] Joel Foster. Plasma vs Optimistic Rollups, August 2020.

[Fou20] Mohamed Fouda. The State of Ethereum L2, July 2020.

[Fra14] Pedro Franco. *Understanding Bitcoin: Cryptography, Engineering and Economics*. WILEY, November 2014.

[FS86] Amos Fiat and Adi Shamir. How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Advances in Cryptology — CRYPTO' 86*, pages 186–194. Springer Berlin Heidelberg, 1986.

[GLSY04] Rosario Gennaro, Darren Leigh, Ravi Sundaram, and William Yerazunis. Batching Schnorr Identification Scheme with Applications to Privacy-Preserving Authorization and Low-Bandwidth Communication Devices. In Pil Joong Lee, editor, *Advances in Cryptology - ASIACRYPT 2004*, pages 276–292, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[Glu19a] Alex Gluchowski. Nearly-zero cost attack scenario on Optimistic Rollup, October 2019.

[Glu19b] Alex Gluchowski. Optimistic vs. ZK Rollup: Deep Dive, November 2019.

[Glu20] Alex Gluchowski. Evaluating Ethereum L2 Scaling Solutions: A Comparison Framework, June 2020.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing - STOC 85*. ACM Press, 1985.

[GV19] Alex Gluchowski and Alex Vlasov. Introducing Matter Testnet, January 2019.

[Kog19] Dima Kogan. Lecture: Proofs of Knowledge, Schnorr's protocol, NIZK, 2019.

[LK19] Avihu Levy and Uri Kolodny. Validity Proofs vs. Fraud Proofs Strike Back, December 2019.

[LK20] Avihu Levy and Uri Kolodny. The Optimistic Rollup Dilemma, October 2020.

[Low20] Rand Low. What is the scalability trilemma?, 2020.

[Mos18] Patricio Mosse. Ethereum's Layer 2 Scaling Solutions, November 2018.

[Mou16] William Mougayar. *The Business Blockchain: Promise, Practice, and Application of the Next Internet Technology*. WILEY, April 2016.

[Nak08] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System, October 2008.

[Nas50] John F. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 36(1):48–49, jan 1950.

[Nit20] Anca Nitulescu. zk-SNARKs: A Gentle Introduction, 2020.

[O'N19] Stephen O'Neal. Who Scales It Best? Inside Blockchains' Ongoing Transactions-Per-Second Race, January 2019.

[Opa18] Ivan Oparin. The First Thread — depicting the moment of divergence in perception of Bitcoin evolution, April 2018.

[Pay19] Artem Payvin. The Data Availability Problem, April 2019.

[PB17] Joseph Poon and Vitalik Buterin. Plasma: Scalable Autonomous Smart Contracts, August 2017.

[Ped91] Torben Pryds Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *Advances in Cryptology — CRYPTO '91*, pages 129–140. Springer Berlin Heidelberg, 1991.

[QG19] Kaihua Qin and Arthur Gervais. An overview of blockchain scalability, interoperability and sustainability. Technical report, EU Blockchain, 2019.

[QQQ+89] Jean-Jacques Quisquater, Myriam Quisquater, Muriel Quisquater, Michaël Quisquater, Louis Guillou, Marie Annick Guillou, Gaïd Guillou, Anna Guillou, Gwenolé Guillou, and Soazig Guillou. How to Explain Zero-Knowledge Protocols to Your Children. In *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 628–631. Springer New York, August 1989.

[Rei16] Christian Reitwießner. zkSNARKs in a Nutshell, December 2016.

[RQ20] Ashwin Ramachandran and Haseeb Qureshi. The Life and Death of Plasma, January 2020.

[Sai19] Vaibhav Saini. Getting Deep Into EVM: How Ethereum Works Backstage, November 2019.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, jan 1991.

[SL19] Sacha Saint-Leger. Istanbul, zkRollup, and Ethereum throughput limits: an analysis, December 2019.

[The20] TheBlock. Ethereum miners are increasing the network's gas limit by 25percent, June 2020.

[Tok20] TokenTerminal. A Primer on Ethereum L2 Scaling Techniques, July 2020.

[VIS19] VISA. Annual Report 2018, January 2019. Page 7.

[VP19] Alexander Vlasov and Konstantin Panarin. Transparent Polynomial Commitment Scheme with Polylogarithmic Communication Complexity, September 2019.

[VS18] Surya Viswanathan and Aakash Shah. The Scalability Trilemma in Blockchain, October 2018.

[Wal20] Petro Wallace. Layer 1 vs Layer 2 : What you need to know about different Blockchain Layer solutions, March 2020.

[Whi18] Barry Whitehat. roll-up, September 2018.

[Wu19] Kimi Wu. ZK Rollup and Optimistic Rollup (En), October 2019.

[WZS19] Xun (Brian) Wu, Zhihong Zou, and Dongying Song. *Learn Ethereum*. Packt Publishing, September 2019.

[Yua19] Michael Yuan. *Building blockchain apps*. Addison Wesley, Place of publication not identified, 2019.

[Zca20] Zcash. What are zk-SNARKs?, 2020.

A Proof of Pedersen protocol

$$\begin{aligned} g^u * h^v \pmod n &= g^{p+e+x} * h^{q+e+r} \pmod n \\ &= g^p * h^q * (g^x)^e * (h^r)^e \pmod n \\ &= d * (g^x * h^r)^e \pmod n \\ &= d * c^e \pmod n, \end{aligned} \tag{2}$$

which proves that $g^u * h^v \pmod n$ equals $d * c^e \pmod n$.

B Throughput approximation in Python

The following python script is also available on GitHub.

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Oct  2 14:58:29 2020

@author: tobiasschaffner
"""
#Inspired by Karl Floersch's work on OR's throughput: https://gist.github.com/karlfloersch/1bf6ab7871f41e3a5a921c0a007ad5c6


# This function helps us calc how much many transitions & transactions we can fit in a block with some calldata size
def get_cost_with_transition_size(transition_bytes, txs_in_a_transition=1, fixed_gas_cost=0, IS_EIP_2028_INCLUDED=True):
    if not IS_EIP_2028_INCLUDED:
        cd_gas_per_byte = 68
    else:
        cd_gas_per_byte = 16
    gas_per_sstore = 20000
    gas_cost_of_merkle_root = gas_per_sstore
    cd_gas_per_tr = cd_gas_per_byte * transition_bytes
    total_gas_in_block = 12500000

    def sha3_gas(num_bytes):
        num_words = num_bytes // 32
        return 30 + 6 * num_words

    sha3_gas_of_one_hash = sha3_gas(32)
    gas_for_one_set_of_trs = sha3_gas(transition_bytes) + cd_gas_per_tr + sha3_gas_of_one_hash
    # Note: Added `sha3_gas_of_one_hash` gas cost per tr (transition) because merkle tree hash cost is 2x number of leaves
    total_trs_in_block = ((total_gas_in_block - gas_cost_of_merkle_root - fixed_gas_cost) / gas_for_one_set_of_trs)
    print('total transitions in block:', total_trs_in_block)
    print('gas per transition:', total_gas_in_block / total_trs_in_block)
    eth_block_time = 14
    print('avg txs per second', total_trs_in_block//eth_block_time*txs_in_a_transition)

state_root = 32  # bytes


#----------------------------------------
```

```
39
40
41
42  # ERC20 Transfer with ECDSA signature before Istanbul Fork - Note IS_EIP_2028_INCLUDED is
        changed to False
43  # Note despite being in a pre-Istanbul set-up, total_gas_in_block = 12.5M instead of 8M is used.
        Parameter can be adjusted, tho.
44  ecdsa_sig = 65  # bytes. Note from the signature we can determine the sender
45  recipient_address = 3  # bytes
46  amount = 7  # bytes
47  num_txs_in_transition = 15
48  transition_size = ((ecdsa_sig + recipient_address + amount) * num_txs_in_transition) +
        state_root
49  print('size,', transition_size)
50  # Calculate
51  print('\nECDSA␣ERC20␣Transfers␣b4␣Istanbul')
52  get_cost_with_transition_size(transition_size, num_txs_in_transition, 0, False)
53
54
55
56  # ERC20 Transfer with ECDSA signature after Istanbul Fork - Note IS_EIP_2028_INCLUDED is changed
        to True
57  ecdsa_sig = 65  # bytes. Note from the signature we can determine the sender
58  recipient_address = 3  # bytes
59  amount = 7  # bytes
60  num_txs_in_transition = 15
61  transition_size = ((ecdsa_sig + recipient_address + amount) * num_txs_in_transition) +
        state_root
62  # Calculate
63  print('\nECDSA␣ERC20␣Transfers')
64  get_cost_with_transition_size(transition_size, num_txs_in_transition, 0, True)
65
66
67
68  # ERC20 Transfer with BLS signature before Istanbul Fork - Note IS_EIP_2028_INCLUDED is changed
        to False
69  # Note despite being in a pre-Istanbul set-up, total_gas_in_block = 12.5M instead of 8M is used.
        Parameter can be adjusted, tho.
70  sender_address = 3  # bytes
71  recipient_address = 3  # bytes
72  amount = 7  # bytes
73  num_txs_in_transition = 15
74  transition_size = ((sender_address + recipient_address + amount) * num_txs_in_transition) +
        state_root
75  gas_for_bls_sig = 113000
76  # Calculate
77  print('\nBLS␣ERC20␣Transfers␣b4␣Istanbul')
78  get_cost_with_transition_size(transition_size, num_txs_in_transition, gas_for_bls_sig, False)
79
80
81
82  # ERC20 Transfer with BLS signature after Istanbul Fork - Note IS_EIP_2028_INCLUDED is changed
        to True
83  sender_address = 3  # bytes
84  recipient_address = 3  # bytes
85  amount = 7  # bytes
86  num_txs_in_transition = 15
87  transition_size = ((sender_address + recipient_address + amount) * num_txs_in_transition) +
        state_root
88  gas_for_bls_sig = 113000
89  # Calculate
90  print('\nBLS␣ERC20␣Transfers')
91  get_cost_with_transition_size(transition_size, num_txs_in_transition, gas_for_bls_sig, True)
```