ZKP Computation Verification:
Scaling Ethereum using Zero-Knowledge Rollups

DD2489 Programmable Society

Vivi Andersson
vivia@kth.se

March 27, 2024

I certify that generative AI, incl. ChatGPT, has not been used to write this essay. Using generative AI without permission is considered academic misconduct.

1 Introduction

In 2008, blockchains as a concept was introduced with the release of Satoshi Nakamoto's paper suggesting a peer-to-peer cash system called Bitcoin. As an infrastructure, blockchains would provide a decentralised and completely trustless system for transactions using cryptographic proofs [1]. Since 2008, various blockchains have emerged, and their usage has extended beyond transacting cryptocurrencies. Today, applications of blockchain technology range from tamper-resistant healthcare recording, digital asset tokenisation, and decentralised governance systems, to decentralised social networking [2].

Since 2008. the number of transactions on blockchains and the sizes of the blocks (i.e. gathered transactions) have increased [3]. For Ethereum, the increase in the number of users has resulted in slower finalisation of transactions and expensive transaction fees, which testifies to a scalability issue [4]. For a wide adoption of Ethereum, scalability is essential. However, considering the tradeoffs in the design goals of decentralisation, security and scalability makes this a difficult pursuit, a problem Ethereum's founder Vitalik Buterin has described as the "Blockchain Trilemma" [5].

A decentralised and secure blockchain requires all nodes running the protocol to verify every transaction that is processed, which is a resource-intensive action and limits the possibility of scalability. One approach to resolving this is to increase the power of the hardware running the protocol, but this would raise the requirements for participating in the system, sacrificing decentralisation. Another approach which does not surrender decentralisation is to send more transactions per second, but this approach loses security due to the increased probability of attacks due to network latency [6]. As Ethereum faces challenges of network congestion and increased hardware requirements for running a node [6], finding viable options for scaling the network is important for surviving in a surging blockchain ecosystem.

2 Terminology

A blockchain is a distributed peer-to-peer database that stores data on a chain in groups called *blocks*. The data is sent in *transactions*, shared among peers running the protocol, known as *nodes*. Transactions on the Ethereum network mainly consist of transfers of cryptocurrency or deployment and interaction with computer programs known as *smart contracts*. Each subsequent block includes a hash of the previous block in itself, creating an *immutable chain* of blocks. Nodes that run a blockchain protocol must agree on a state, which is handled through the *consensus mechanism*. The Ethereum protocol adopts *Proof-of-Stake* consensus, which requires creators of blocks, nodes known as *validators*, to stake ether (ETH), the native currency of Ethereum. A transaction is considered verified and finalised when it is written on the chain [6].

3 Scalability

The two main goals of scaling a blockchain are to increase the *throughput* of transactions and the transaction *finality speed*. Increasing the transaction throughput has the possibility of lowering the fees for users, and is commonly measured in transactions per second (TPS) [4] [7]. The key challenge of the Blockchain Trilemma has resulted in various approaches to blockchain scalability, which can be categorised as *on-chain* or *off-chain* approaches [4].

3.1 On-Chain Scaling

On-chain scaling solutions, also known as layer 1 scaling, require a change of the protocol the main chain (such as Ethereum Mainnet) runs on [4]. For layer 1 scaling, the performance is enhanced by changing parameters such as block sizes and time, or the chain consensus mechanism [7]. A central on-chain scaling approach for Ethereum is called *sharding*, which is the act of splitting up a database into smaller fragments. For a blockchain, sharding entails splitting the responsibility of different shards for validators [4].

3.2 Off-Chain Scaling

Off-chain scaling approaches operate separately from the main blockchain and require no changes to the underlying protocol. Off-chain approaches for Ethereum such as *sidechains* achieve security from their implementation [4]. Conversely, off-chain scaling techniques called *layer 2 scaling* interact with the main chain and inherit its security. The main, layer 1, Ethereum chain currently can process 15 TPS, which is a problem during high load on the network. Layer 2 scaling approaches aim to increase transaction throughput on the main network by allowing transactions to be conducted off the main blockchain, and only publishing the necessary data on the main chain [7] [8]. For Ethereum, the layer 2 approach of *rollups* is currently the dominating scaling technique [4].

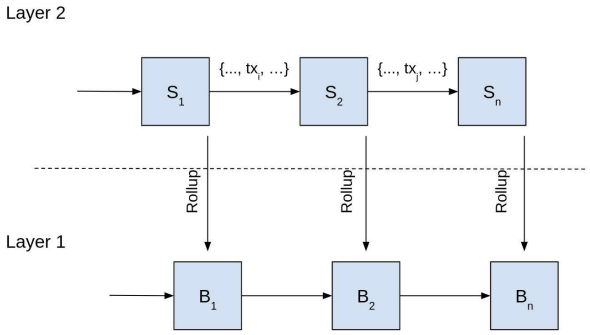

Figure 1: *Layer 1 and 2 rollup interaction [7].*

3.2.1 Layer 2 Rollups

Rollups rely on a separate blockchain which processes transactions off the main chain, which then are "rolled up" into one transaction for the main blockchain (see Fig. 1) [7]. The distribution of transaction execution can lower the transaction fees, for Ethereum, such solutions currently provide about a 3-8 times lower fee [9].

Rollups are smart contracts that reside on the layer 1 chain relaying to the layer 2 chain. Users can transact on the layer 2 chain by depositing funds to the layer 1 contract, which then is exchanged for layer 2 funds. Users then send their transactions to a layer 2 *sequencer* or *aggregator*, benefiting from lower transaction fees, and the transaction is published to the layer 1 chain by the sequencer (see Fig. 2) [7].

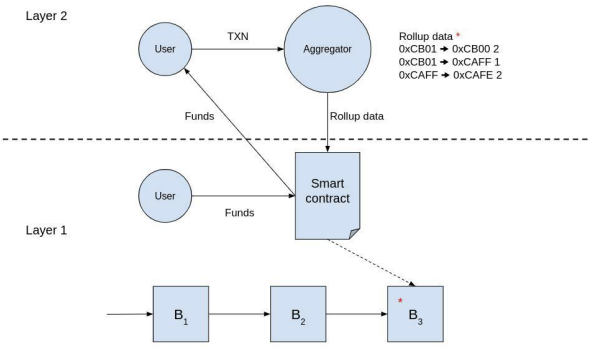

Figure 2: *Rollup contract interaction: a user funds a layer 1 contract, sends a transaction (TXN) to an aggregator, and the data is rolled up to the original contract [7].*

There are currently two main approaches to rollups; *optimistic rollups* and *zero-knowledge (zk) rollups*. The rollups differ in how the transaction data is submitted to the layer 1 chain, and how the data is verified. Zero-Knowledge rollups run any computations off the chain and send a *validity proof* to the layer-1 chain, whereas optimistic rollups assume valid transactions, and only generate *fault proofs* in case of suspected fraud [8].

This essay will focus on the layer 2 zero-knowledge rollup approach to scaling Ethereum.

4 Zero Knowledge Rollups

The first zero-knowledge proof was first shown in the 1989 paper "The Knowledge Complexity of Interactive Proof Systems" by Shafi Goldwasser, Silvio Micali and Charles Rackoff. In this type of proof, the idea is to not convey any more information about a proof than the correctness of it. In the proof scheme, a *prover* tries to convince a *verifier* of a claim, with the help of a *witness*. The verifier accepts or rejects the claim [10]. For blockchains, the zk proofs rely on being non-interactive [7].

4.1 State Changes

The state of the layer 2 rollup is stored in a Merkle tree, which allows for efficient storage on chain. The layer 1 contract for zk rollups holds the *root* of this Merkle tree, which is updated correspondingly after a new batch of transactions in the rollup. The state update to the layer 1 contract consists of a summary of the required state changes and the zero-knowledge validity proof for the transactions [11].

4.2 Validity Proofs

Another *verifier contract* also resides on the layer 1 network, which verifies the submitted proofs in the state updates (see the Appendix for an example of a smart contract code implementation of the verifier, used by the zk rollup chain Loopring [12]). The validity proof proves the state changes that are submitted are correct, without the need to supply the data itself. Thus, zk rollups can prove valid changes on the layer-1 blockchain without storing the actual data on this chain [11].

In 2018, a GitHub user under the name "Barry Whitehat" proposed a layer 2 scaling approach for Etereuem called roll up, using the approach of Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge (zk-SNARK), which was picked up by Ethereum [13]. Today, the validity proofs used for Ethereum zk rollups are either zk-SNARKs or zk-STARKs, i.e. Zero-Knowledge Scalable Transparent Argument of Knowledge [11].

5 State of the Art

5.1 zk-SNARKs

Zk-SNARKs were first introduced in 2012 by Nir Britansky et al, providing succinct proofs and constant time verification, making this type of proof effective for blockchain applications [14] [15]. A trusted set-up phase creates the common reference strings (CRS), which is a central part of the zk-SNARK protocol. The method relies on Quadratic Arithmetic Programs (QAPs) and elliptic curve cryptography (ECC) to construct the proofs [14].

5.2 zk-STARKs

Zk-STARKS improve on the zk-SNARK approach by enhancing scalability and transparency [16]. This is achieved by eliminating the need for a trusted set-up phase and instead relying on publicly verifiable randomness. The scalability is derived from more efficient verification, however, this comes at the cost of larger proof sizes. Another advantage of zk-STARKS is its resistance to quantum computing attacks, by addressing a vulnerability in the ECC process included in the zk-SNARK protocol [11].

6 Discussion

Zk rollups offer layer 2 scaling solutions that enable fast interaction with the main chain, ensuring both security and reduced transaction fees for users. Future estimates for Ethereum suggest zk rollups could potentially lower fees by 40-100 times compared to the current cost [9].

In contrast to optimistic rollups, zk rollups finalise faster due to transactions arriving with verification of correctness. However, this speed comes at the expense of increased costs, as zk rollups require computational proofs [13]. For zk-STARKS which scales better, these proofs become even larger.

A potential drawback of zk rollups is the risk of centralisation, derived from the specialised hardware needed for validity proof generation. This poses a challenge to the decentralised nature of blockchains. Furthermore, the reliance on a trusted set-up process, such as in zk-SNARKs, introduces security concerns. Any compromise in this process could compromise the overall security of the rollup [11].

Additionally, given the early stage of layer 2 scaling, inherent risks exist such as zero-day attacks. For example, the bridge connecting layer 1 and layer 2 introduces a potential attack vector [8].

7 Conclusion

In conclusion, we have seen the scalability issue present for the Ethereum blockchain, and possible solutions at hand. Zk rollups rely on precise and complex cryptographic proof techniques, which has several benefits, but with tradeoffs in accessibility and decentralisation due to its complex nature. Nonetheless, as one of the leading scaling approaches for Ethereum, zk is expected to substantially increase the throughput of transactions in the future [9]. Yet, as stressed by Ethereum [6], the complexity of the Blockchain Trilemma requires a diverse set of solutions to scaling, including both on-chain and off-chain approaches, which is why it is important to consider zk rollups only being part of the solution to a scalable Ethereum infrastructure.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Bitcoin.org*, 2008. available at: https://bitcoin.org/bitcoin.pdf.

[2] J. Abou Jaoude and R. George Saade, "Blockchain applications – usage in different domains," *IEEE Access*, vol. 7, pp. 45360–45381, 2019.

[3] A. Hafid, A. S. Hafid, and M. Samih, "Scaling blockchains: A comprehensive survey," *IEEE Access*, vol. 8, pp. 125244–125262, 2020.

[4] Ethereum, "Scaling," 2023. https://ethereum.org/en/developers/docs/scaling/ (retrieved 2023-12-10).

[5] V. Buterin, "The limits to blockchain scalability," 2021. https://vitalik.ca/general/2021/05/23/scaling.html (retrieved 2023-12-10).

[6] Ethereum, "The ethereum vision," 2023. https://ethereum.org/en/roadmap/vision/ (retrieved 2023-12-10).

[7] L. T. Thibault, T. Sarry, and A. S. Hafid, "Blockchain scaling using rollups: A comprehensive survey," *IEEE Access*, vol. 10, pp. 93039–93054, 2022. doi: https://doi.org/10.1109/ACCESS.2022.3200051.

[8] Ethereum, "Layer 2," 2023. https://ethereum.org/en/layer-2/ (retrieved 2023-12-11).

[9] Ethereum, "Scaling ethereum," 2023. https://ethereum.org/en/roadmap/scaling/ (retrieved 2023-12-11).

[10] S. Goldwasser, S. Micali, and C. Rackoff, *The Knowledge Complexity of Interactive Proof-Systems*, p. 203–225. New York, NY, USA: Association for Computing Machinery, 2019. doi: https://doi.org/10.1145/3335741.3335750.

[11] Ethereum, "Zero-knowledge rollups," 2023. https://ethereum.org/en/developers/docs/scaling/zk-rollups/ (retrieved 2023-12-11).

[12] Loopring, "Ethereum's first zkrollup layer 2," 2023. https://loopring.io/#/ (retrieved 2023-12-11).

[13] C. Capital, "Zk-rollups landscape overview," July 2023. https://research.cryptomeriacapital.com/Cryptomeria_Capital_ZK_Rollups_Landscape_Overview_July_2023.pdf (retrieved 2023-12-11).

[14] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer, "From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, (New York, NY, USA), p. 326–349, Association for Computing Machinery, 2012. doi: https://doi.org/10.1145/2090236.2090263.

[15] T. Chen, H. Lu, T. Kunpittaya, and A. Luo, "A review of zk-snarks," 2023.

[16] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Scalable, transparent, and post-quantum secure computational integrity." Cryptology ePrint Archive, Paper 2018/046, 2018. https://eprint.iacr.org/2018/046.

Appendix A

An excerpt of the computation verification used by layer 2 zk rollup Loopring, based on the zk toolbox Zokrates. Available at https://github.com/Loopring/ethsnarks/ and original at https://github.com/JacobEberhardt/ZoKrates.

```
function Verify ( uint256[14] memory in_vk, uint256[] memory vk_gammaABC,
uint256[8] memory in_proof, uint256[] internal view returns (bool)
{
    uint256 snark_scalar_field =
    21888242871839275222246405745257275088548364400416034343698204186575808495617;

    require( ((vk_gammaABC.length / 2) - 1) == proof_inputs.length );
    // Compute the linear combination vk_x
    uint256[3] memory mul_input;
    uint256[4] memory add_input;
    bool success;
    uint m = 2;
    // First two fields are used as the sum
    add_input[0] = vk_gammaABC[0];
    add_input[1] = vk_gammaABC[1];
    // Performs a sum of gammaABC[0] + sum[ gammaABC[i+1]^proof_inputs[i] ]
    for (uint i = 0; i < proof_inputs.length; i++)
    {
        require( proof_inputs[i] < snark_scalar_field );
        mul_input[0] = vk_gammaABC[m++];
        mul_input[1] = vk_gammaABC[m++];
        mul_input[2] = proof_inputs[i];
        assembly {
            // ECMUL, output to last 2 elements of `add_input`
            success := staticcall(sub(gas, 2000), 7, mul_input, 0x80, add(add_input, 0x40), 0x60)
    }
    require( success );
    assembly {
        // ECADD
        success := staticcall(sub(gas, 2000), 6, add_input, 0xc0, add_input, 0x60)
    }
    require( success );
}
```