polygon zkEVM

Knowledge Layer

Concepts

L2 Scaling Strategies

Version: 6e9af772d5eb8dd4087313d0f660c9d3f77fa64c

January 9, 2024

1 Introduction to Layer 2

L2 layers are be established on top of the L1 infrastructure. These L2 layers have the authority to dictate the type of state processing that support. For example, a payment system with simple transactions, a token transfer system or the EVM execution capability. Moreover, L2 layers have the autonomy to define their utilization of the underlying L1 architecture, encompassing two pivotal aspects: the L1 execution layer's usage and the interaction with the data shards (when available in production). Regarding the current situation of data sharding, L1 data shards specification is still not present in mainnet, in particular, the **EIP 4484**, which provides its specification, is currently on the "review stage". For this reason, currently, scaling solutions use only the execution layer (i.e. the beacon chain) for making the L2 data available.

For the sake of illustration, let's consider an instance of an L2 layer denoted by $L2_x$. In this scenario, $L2_x$ **transactions** form the foundational operations within the $L2_x$ state. These transactions shape the state progression of $L2_x$. Figure 1 illustrates how the different $L2_x$ states change with the execution of new sets of transactions.

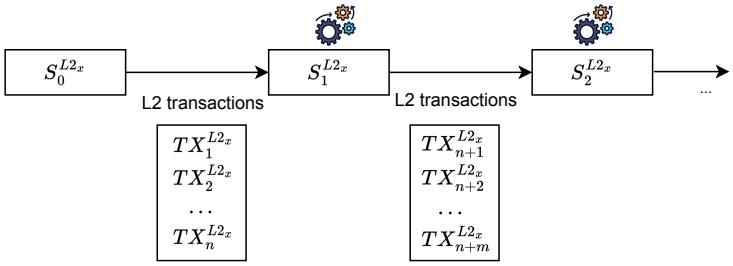

Figure 1: L2 State evolution via the execution of sets of L2 transactions.

Within this context, several questions naturally emerge:

(a) **Transaction Dissemination**: *How do users initiate and direct $L2_x$ transactions, and who are the receivers of these transactions?*

(b) **Transaction Data Availability**: *Is there a mechanism in place to make L2 transactions publicly accessible, and if so, what methods are employed to achieve this transparency within the $L2_x$ ecosystem?*

(c) **Transaction Processing and State Verification**: *Who bears the responsibility for processing L2 transactions, what methodologies govern this processing, and when it can be publicly affirmed that a new state has been accurately computed within $L2_x$?*

(d) **Supported Application Complexity**: *What types of applications find their home within $L2_x$? Does this layer cater primarily to simple processing, or does it have the capacity to facilitate rich state processing environments like the EVM?*

While it's important to note that the answers may not be unique due to the existence of different approaches (each one having its pros and cons), the following points will provide possible responses to all of the questions posed.

1.1 Transaction Reception

In the context of transaction origination and reception, there exists various communication paradigms that may dictate how L2 transactions are disseminated:

- **Unicast**: This dissemination mode involves the creation of a one-to-tone communication channel with a centralized receiver (see Figure 2). In this scenario, L2 transactions are directed towards a specific centralized recipient.

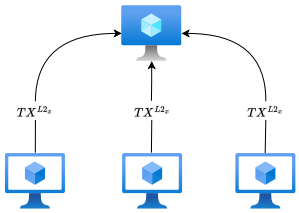

Figure 2: Schema of Unicast communication.

- **Peer-to-peer**: In a peer-to-peer network (see Figure 3), every participant, receives and processes L2 transactions in a decentralized fashion. This approach emphasizes distributed communications, ensuring that all network participants have access to the transactions.

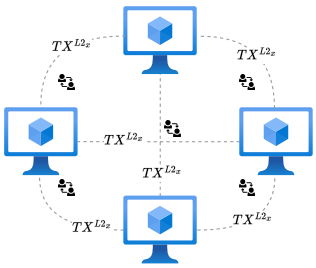

Figure 3: Schema of peer-to-peer communication.

- **Smart Contract**: Another approach to dissemination is through smart contracts, making use of consensus of the L1 execution layer (see Figure 4). These smart contracts operate in a decentralized manner and play a role in managing and processing L2 transactions.

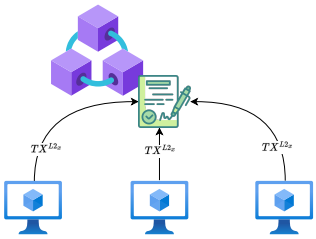

Figure 4: Schema of Smart Contract like communication.

1.2 Transaction Data Availability

To achieve L2 data availability in Ethereum, currently, we can proceed in two ways:

- **Validium Approach**: In this approach (see Figure 5), L1 transactions only contains a cryptographic summary (a hash) of the corresponding L2 transactions. This design does not allow a user to retrieve L2 data from the L1 layer, which is the

public layer. Instead, L2 data is managed by a consortium of trusted entities known as *data managers*. Hence, users must rely on data managers to access L2 data.

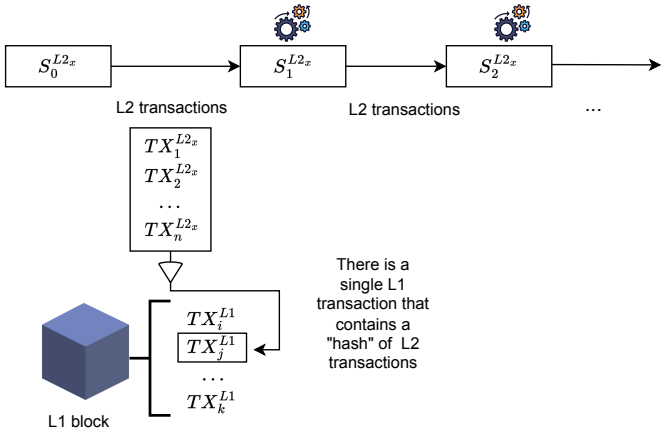

Figure 5: Data Availability with the Validium Approach.

- **Rollup Approach**: This approach involves writing L2 data to the public L1 execution layer, making the posted data publicly accessible to all participants on the blockchain network (see Figure 6). In this case, to provide data availability, we use a single L1 transaction that contains a batch of L2 transactions. This method is called a rollup, because we *roll up* a bunch of L2 transactions in a single L1 transaction.

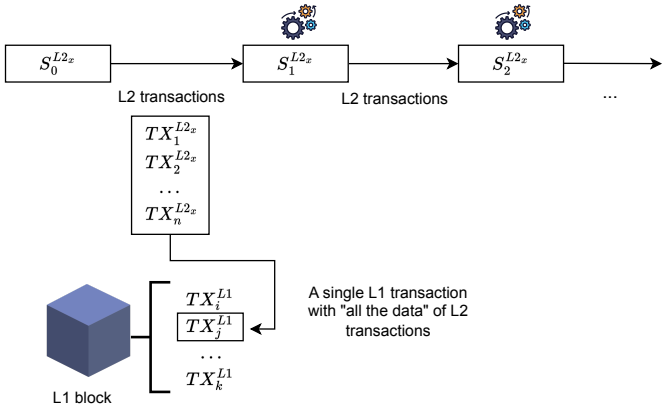

Figure 6: Data Availability with the Rollup Approach.

1.3 Transaction Processing and State Verification

In the context of computing the subsequent L2 state $S_{i+1}^{L2_x}$ based on a set of $L2_x$ transactions and the present state $S_i^{L2_x}$, several distinct approaches can be employed. These approaches encompass: centralized execution, optimistic execution and succinct computation verification.

Centralized Execution: This approach involves centralized execution (see Figure 7), where a central entity, called L2 operator, is responsible for processing L2 transactions and computing the new L2 state. In this approach, the process of finalizing state computation occurs quite rapidly, typically taking only a matter of "seconds or minutes". However, this

approach introduces the significant challenge of how external entities can raise disputes with the L2 operator regarding the accuracy and correctness of an L2 state computation.
Optimistic Execution: This approach is characterized by a trust-first paradigm. In this

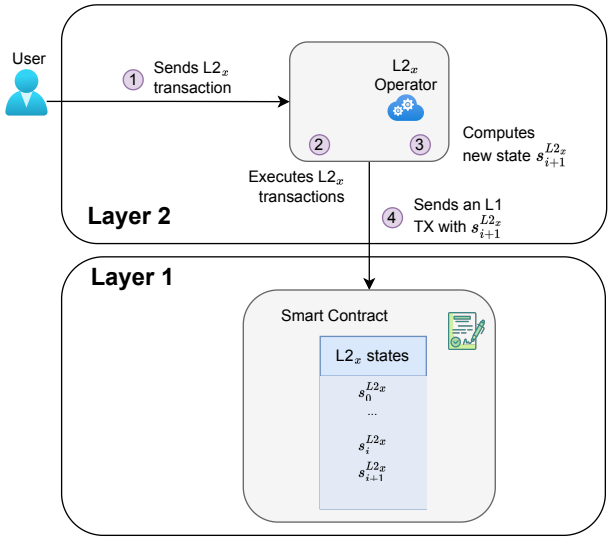

Figure 7: Centralized Execution Schema.

paradigm, the state is computed *optimistically*, that is, assuming it is correct and only in cases of disputes the state is rigorously verified. An Optimistic L2 provides a decentralized execution mechanism that allows disputing the correctness of the L2 state computation. With this approach, there is a (large) period of time to allow anybody to send a **fraud proof** (see Figure 8), proving that the state was wrongly computed (for example, a double spending transaction). Under this model, if those who execute L2 transactions (referred

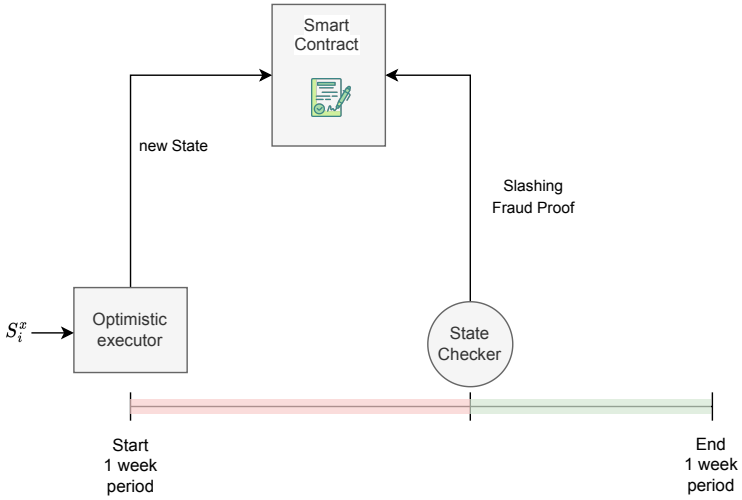

Figure 8: Optimistic Execution Schema of a Fraud Proof.

to as the **optimistic executors**) correctly perform their duties, they stand to earn ETH as a reward for their correct execution. However, in cases where an optimistic executor fails to execute transactions correctly, they face a penalty in the form of slashing and the actor that provides the fraud proof (referred to as the **state checker**) is rewarded for

that task. This approach aims to strike a balance between efficiency and security.

Succinct Computation Verification (Zero-Knowledge Technology): The third approach leverages zero-knowledge technology to enable succinct computation verification. In this method, transactions are processed and verified in a manner that allows third parties to efficiently confirm the validity of state computations without needing to re-execute them. In the succinct execution verification model, instead of an optimistic executor, we have an execution prover (see Figure 9).

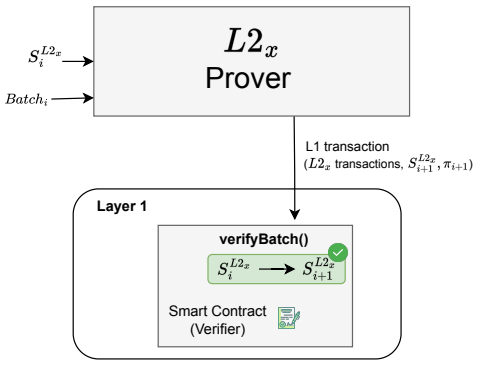

Figure 9: Succinct Computation Verification Schema.

The **prover** is the entity responsible for generating a proof that shows that the new L2 state is correctly computed from the previous state and a given set of L2 transactions (which is called a **batch**) by using zero-knowledge techniques. The **verifier** role, which have to confirm the validity of the proofs, is assumed by a smart contract located in the the Layer 1 that is also responsible for storing the set of states. With modern ZKP techniques, the proof size is exceptionally succinct, consisting of just a few bytes. This compactness reduces data storage and transmission requirements. Verification time is notably brief, typically of the order of microseconds (μs). This inherent properties of the proof enable a smart contract to perform successful verifications.

When the verifier smart contract executes the L1 transaction that verifies a batch of L2 transactions, we say that the next state, S_{i+1}^x, is **consolidated**. With this approach, the state computation is quickly considered final since it takes only "seconds or minutes" to generate and validate the proof.

The **prover** is typically hosted on a cloud service, allowing for a elastic demand of resources. Provers require a significant amount of computational resources to generate cryptographic proofs, primarily in terms of RAM and CPU.

However, given that all L2 data is shared on L1, it naturally raises the question of whether we are actually achieving scalability through rollups based on succinct verification. The affirmative response stems from the fact that the smart contract execution resources for verifying a proof are much lower than executing individual L2 transactions within a batch. In fact, for Layers 2, the majority of the cost comes from the data availability which is written in L1. Most L2 solutions try to improve data availability costs by using succinct verification of compressed data and they will use data sharding when available for production.

As a final remark, it's important to recognize that the selected method can significantly impact the performance, security, and decentralization characteristics of the L2 ecosystem. Hence, careful consideration and alignment with specific use cases are essential when selecting the appropriate approach.

1.4 Summary of L2 Scaling Solutions

Scalability solutions for blockchains can be categorized along two main dimensions:

- **Data availability**: Whether the data from the L2 chain is available on-chain (in L1) or off-chain (e.g. data managers).
- **Mechanism to ensure correctness**: The methods employed to guarantee the correctness and validity of the L2 chain's state.

The table below shows an overview of the scaling solutions:

	Validity Proof	Fraud Proof
Data on-chain	zkRollup	Optimistic Rollup
Data off-chain	zkValidium	Plasma

One noteworthy feature of the zkValidium approach is its reliance on ZKP technology to ensure that even if L2 operators withhold data from users, the cryptographic validation mechanisms guarantee the correct processing of transactions (see Figure 10). This means that, despite the lack of visibility into individual L2 transactions users can be guaranteed about correctness of the L2 state computations. This approach is more cost-effective compared to the zkRollup approach, that consists on writing data directly to the L1 layer.

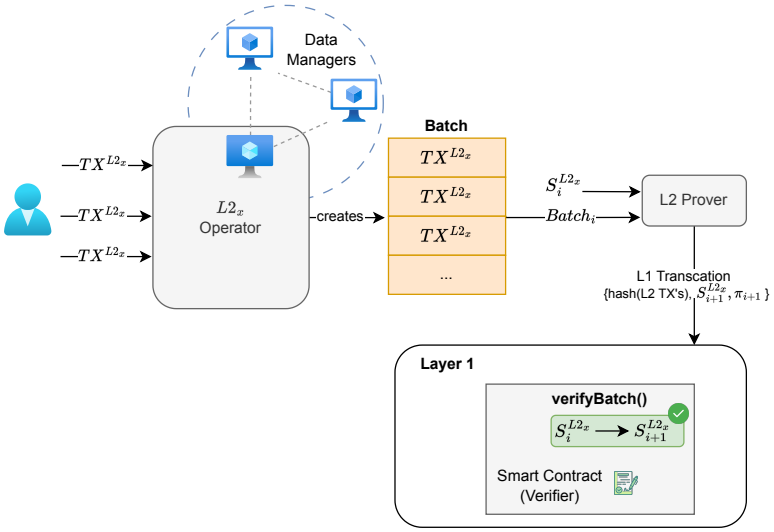

Figure 10: zkValidium Batch Verification Process.

1.5 Supported Application Complexity

Within Layer 2 solutions, the scope of supported functionalities can be categorized into two primary types:

- **Simple Processing**: This category encompasses solutions that involve the simple transfer of digital assets, such as tokens or payments, with simple processing requirements (see Figure 11).

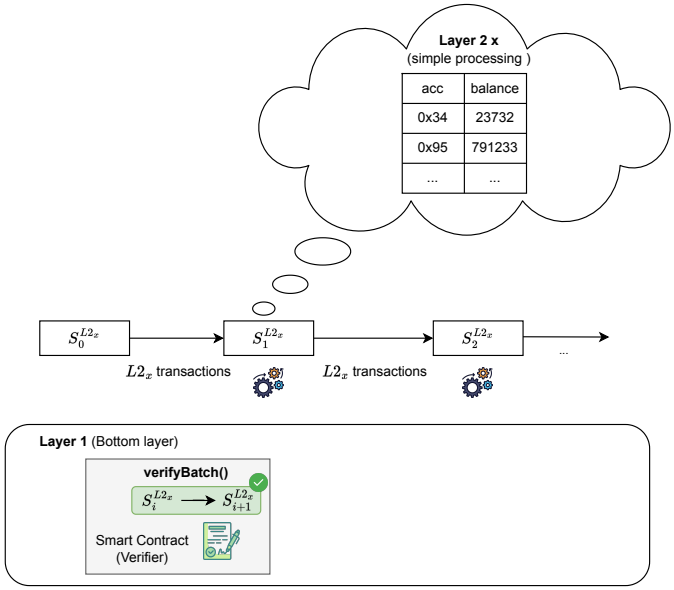

Figure 11: Simple Processing Schema.

- **General-Purpose Execution Virtual Machine**: This category extends the capabilities of L2 solutions to encompass a broader spectrum of functionalities (see Figure 12). A remarkable example in this category is the implementation of an L2 that supports the EVM processing which is a rich transaction processing environment that allows users to program their desired logic for transaction processing via small programs called smart contracts.

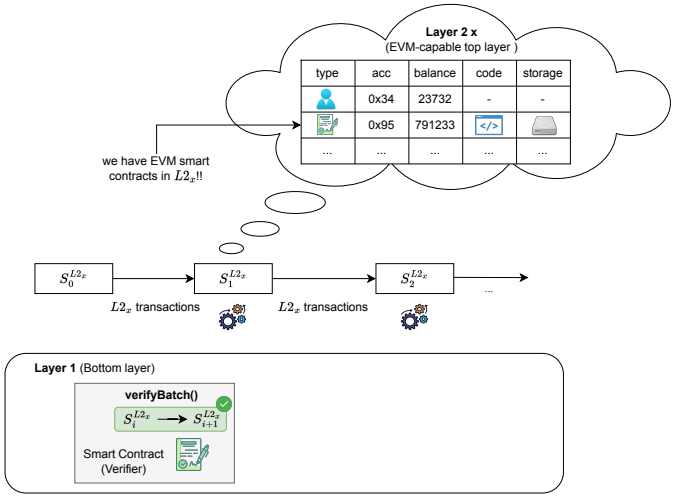

Figure 12: EVM-like Processing Schema.

1.6 Answering the L2 Design Questions for the Polygon zkEVM

- **How users send L2 transactions and who receives them?**

  The zkEVM uses unicast to let users send their transactions to a zkEVM node via RPC calls. The zkEVM also enables posting L2 transactions via a function in a

smart contract as an anti-censorship measure (called “forced batches”).

- **How L2 transactions are made publicly available (if so)?**

  The zkEVM is a rollup and L2 data is available in the L1 execution layer (beacon chain).

- **Who process the L2 transactions and how, and, when it is publicly considered that a new state is correctly computed?**

  In zkEVM, there is presently an aggregator node responsible for generating proofs stating the correct execution of Layer 2 transactions. It’s important to note that this node is prevented from engaging in fraudulent activities due to the presence of succinct computation verification, which utilizes zero-knowledge technology.

- **What type of applications the L2 supports? The processing is simple or rich?**

  zkEVM is rich processing since it is an EVM-based L2.