A Public Dataset For the ZKsync Rollup*

Maria Inês Silva[1,4], Johnnatan Messias[2], and Benjamin Livshits[3]

[1] NOVA Information Management School
[2] MPI-SWS
[3] Imperial College London
[4] Matter Labs

Abstract. Despite blockchain data being publicly available, practical challenges and high costs often hinder its effective use by researchers, thus limiting data-driven research and exploration in the blockchain space. This is especially true when it comes to Layer-2 (L2) ecosystems, and ZKsync, in particular. To address these issues, we have curated a dataset from 1 year of activity extracted from a ZKsync Era archive node and made it freely available to external parties. We provide details on this dataset and how it was created, showcase a few example analyses that can be performed with it, and discuss some future research directions.

1 Introduction

The blockchain ecosystem is built on the principles of decentralization and transparency. However, a significant challenge remains for end users and non-technical researchers: *blockchain data is not easily accessible*. This challenge hinders the widespread adoption of blockchain technology, which should be more easily accessible.

Currently, blockchain data can be obtained by deploying an archive node in the case of chains based on Ethereum Virtual Machine (EVM) [8,12,30,33], or a full node for Bitcoin [11]. However, this requires high bandwidth and a high-speed storage machine to keep the blockchain fully synced. In other words, individual deployment of archive nodes is not a practical solution. Additionally, users can gather the data through Remote Procedure Calls (RPCs) providers. However, collecting data this way can be challenging for some non-technical users and researchers and is usually costly. Alternatively, users can depend on external data sources such as Etherscan, Arbiscan, Dune, ZettaBlock, Nansen, and similar platforms. Although they may provide an interesting solution, they may prove costly in the long run and do not meet the requirements of specific end-users, particularly those engaged in research that relies on quickly and cheaply accessible data.

We believe that anyone requiring blockchain data should have a straightforward and hassle-free way to access it without worrying about infrastructure or

* MI. Silva and J. Messias contributed equally to this work. Most of this work was performed while the authors were at Matter Labs. The authors also acknowledge the help of Igor Borodin from Matter Labs and his assistance in hosting the dataset.

hardware. This data has significant value for research purposes, such as alerting and measurement analysis [20,25], Airdrop designs and analyzes [15,29], analyzing Maximal-Extractable Value (MEV) [18,19,27,39,42,45], Automated Market Maker (AMM) [16,40], and potentially improving the adoption of particular chains such as ZKsync and other Layer-2 blockchains (L2s) rollups within the research scope [26]. In addition, it can contribute to the growth of these ecosystems as more researchers become involved in the field.

In a recent talk, a company named Paradigm emphasized that to empower the blockchain community with the necessary data analysis capabilities, we must make blockchain data more accessible [6]. By ensuring that these data are available affordably, quickly, and with minimal effort, we can effectively address the challenge of data unavailability in our ecosystem. They introduced a novel EVM blockchain node developed in Rust, known as *Reth*, that can be applied to EVM-compatible blockchains. Paradigm provided an endpoint to their archive node, allowing anyone to make requests to their node. Similar initiatives have been provided by Matter Labs and other companies interested in making blockchain data more accessible [8,30,33]. Nevertheless, users would need to build high-efficiency code to gather all the necessary data, which could invalidate this process for some users due to network delays or lack of coding skills.

More recently, Paradigm [35,36], BigQuery [14] and other research groups [25,27] have made blockchain data accessible and available by providing them in an easy-to-download and-load schema. In that sense, we followed through and decided to make a ZKsync dataset fully available and accessible to any user or researcher. Therefore, we provide 1 year of ZKsync data covering the period of February 14$^{\text{th}}$, 2023, (block 1) and March 24$^{\text{th}}$, 2024, (block 29,710,983). More details of the dataset are available in Section 2.

1.1 Why ZKsync Era Data?

Launched in March 2023, ZKsync Era is a L2 scaling solution for the Ethereum blockchain that utilizes Zero-Knowledge Proof (ZKP) for efficient transaction processing. ZKsync Era is also an EVM-based rollup, ensuring compatibility with existing Ethereum smart contracts. In July 2024, ZKsync Era ranked among the top five ZKP chains with a Total Value Locked (TVL) estimated at 1.23 billion USD [23]. This scaling solution improves Ethereum by processing transactions in batches using ZKP, which helps maintain low transaction fees and encourages user participation.

In this sense, rollups are a key strategy to scale the Ethereum ecosystem [13]. In fact, these L2 chains have become an important part of the ecosystem in the last few years by pushing new innovations and absorbing a significant portion of user activity. At the same time, there are still many unexplored questions when it comes to these L2 blockchains, and existing research on them is fairly limited.

Given the growing role of L2 chains in the Ethereum ecosystem, we believe there is an opportunity for researchers to conduct research and expand our understanding of these L2 blockchains, such as ZKsync. By making a 1-year ZKsync

Era data easily available to external groups, we hope to advance ZKsync-related research and knowledge about L2 chains, more generally.

1.2 Contributions

▶ **Public Availability of ZKsync Data.** To facilitate scientific use of our collected ZKsync Era dataset, we have made it available in a public GitHub repository [4]. This consists of a one-year dataset containing information regarding blocks, transactions, receipts, and logs. Dataset details are presented in Section 2.

▶ **Facilitating Research and Analysis.** We demonstrated potential applications of this dataset in research and advocating for the adoption of specific blockchain technologies like ZKsync and L2 rollups in general in Section 3.

▶ **Practical Implications for Users and Researchers.** We address challenges associated with data gathering via RPCs and external platforms like Etherscan, Arbiscan, and others, which may be slow and costly. We also provide an easy-to-download and easy-to-load dataset of ZKsync Era, along with Jupyter notebooks to facilitate this onboarding process [4]. Table 1 describes the code utilized to load and process our dataset. All the analyses conducted in this paper can be easily reproduced using our code, which will be available in a GitHub repository. See Sections 2 and 3 for details.

1.3 Paper Organization

This paper is organized as follows. Section 2 details our dataset and its data schema, together with the necessary background to understand the context of blockchain data in general. Section 3 provides examples of analyses that can be achieved using this dataset, such as regarding transaction fees and gas usage, events derived from transaction logs, token swaps, and more. Furthermore, Section 4 discusses open problems and future directions for which this dataset could be useful. Finally, we present the conclusion of our work in Section 5.

2 Data Schema and Processing

In this section, we present our ZKsync dataset in detail. The dataset covers the period of February 14$^{\text{th}}$, 2023, (block number 1) and March 24$^{\text{th}}$, 2024, (block number 29,710,983) corresponding to 1 year of data. It contains 327,174,035 transactions and 1,631,772 contracts deployed during this time period which triggered 2,044,221,151 events on-chain. Transactions were issued by 7,322,502 unique users. This dataset enables researchers or blockchain enthusiasts to explore all the activities that occurred in the ZKsync Era since its deployment.

We gathered our dataset from a ZKsync Era archive node as raw data. This data consists of all the information regarding blocks, transactions, receipts, and logs. Then, we conducted a pre-processing step to allow anyone to use the

Table 1: Description of the code used for analysis in this paper. These notebook files are available in our GitHub repository [4] in the directory *./zksync-data-dump/notebooks/* and show how to interact and process our ZKsync dataset.

Notebook file	***Description***
01-zksync-data.ipynb	It computes the basic statistics of the dataset and provides analyses used in Section 2. We use four main sources of data: *blocks*, *transactions*, *transaction receipts*, and *logs*.
02-data-exploration-fees.ipynb	It analyses gas usage and transaction fees for ZKsync used in Section 3.1. We use two main sources of data: *blocks* and *transaction receipts*.
03-data-exploration-contracts.ipynb	It analyzes the contract deployment and events triggered on ZKsync described in Section 3.2. We use one main source of data: *transaction logs* but also load *blocks* data to extract timestamps information.
04-data-exploration-swaps.ipynb	It analyzes the swap events on ZKsync described in Section 3.3. We use one main source of data: *transaction logs* but also load *blocks* data to extract timestamps.

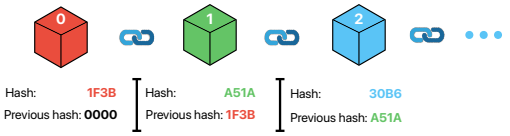


Fig. 1: *Illustration of blocks connected to each other forming the blockchain.*

dataset. This step consists of formatting the data in a parquet format that can be easily accessible through well-known libraries available in Python, for example, Pandas [34] and Polars [38]. Due to the high volume of data (i.e., around 200 GB), we focus on using Polars for better processing, memory management, and *Lazy evaluation* [22,37], allowing straightforward data processing on a local laptop. Next, we discuss each of the different data types separately.

2.1 Blocks

Blocks are sequential units of data within a blockchain, each identified by a unique hash. They contain a list of transactions, metadata such as timestamps, and the hash of the previous block (*parentHash*), which links them in a chain back to the genesis block (block number 0) as shown in Figure 1. This chain of blocks forms the blockchain. Blocks ensure transaction security, network consensus, and efficient data storage and processing within blockchain networks. In our dataset, blocks should be sorted according to the attribute *block number* to maintain the correct order of the blocks on the blockchain.

2.2 Transactions

Transactions are digital interactions that involve the transfer of assets, the recording of data, or the execution of smart contracts between parties on blockchains like those based on EVM. Each transaction is initiated by a user, authenticated through cryptographic signatures, and sent to a decentralized network of

nodes for validation. Once verified, transactions are grouped into blocks and added to the blockchain via a consensus mechanism, ensuring that they are secure, immutable, transparent, and free of intermediaries, forming the core of the blockchain system.

In rollups, such as ZKsync, transactions are aggregated and processed off the underlying blockchain (e.g., Ethereum, a Layer-1 blockchain (L1)) to enhance scalability and reduce costs. Rollups bundle multiple transactions into a single batch, which is then submitted to the underlying blockchain as one transaction. This method reduces the load on the underlying chain while ensuring transaction security and finality through cryptographic proofs, such as ZKP used by ZKsync or validity checks. By processing transactions off-chain and periodically committing the results to the underlying chain, rollups improve throughput and efficiency without compromising the security and decentralization of the blockchain.

Transactions are identified by a unique transaction hash. When issuing a transaction, the user needs to specify parameters such as the recipient address (which can also be a smart contract and the functions the user wants to call), the number of tokens to transfer, the gas price, and the gas limit. The gas price represents the fee the user is willing to pay per unit of gas, while the gas limit is the maximum amount of gas the user is willing to consume for the transaction, a mechanism introduced to prevent infinite loops or excessive resource consumption.

In our dataset, transactions should be sorted by the *blockNumber* and *transactionIndex* attributes to maintain the correct order of the transactions on the blockchain.

2.3 Transactions Receipts

Besides the transaction data, our dataset also contains a table of transaction receipts. These transaction receipts provide a comprehensive summary of the outcome and effects of a transaction once it is processed and included in a block. They include the transaction hash, block number, and block hash to identify and verify the transaction, along with the sender (*from*) and recipient (*to*) addresses. The receipts also detail the cumulative gas used by the transaction and all preceding transactions in the block, the actual gas used by the specific transaction, and the final gas price paid by the user. By multiplying the actual gas used by the gas price, we have the actual transaction fee the user paid.

Additional information includes *logs* for event logging (discussed next), the transaction status (success or failure), and the effective gas price paid. These receipts are crucial for users and developers to understand, audit, and interact with transactions and smart contracts on the blockchain. For example, they provide essential elements for analyzing, monitoring, and verifying transaction fees spent by users.

Similarly to transactions, in our dataset, transaction receipt data should be sorted by attributes *blockNumber* and *transactionIndex* to maintain the correct order of the transactions on the blockchain.

2.4 Transactions Logs

In this section, we discuss the attributes of the transaction logs data in the ZKsync dataset in detail. Transaction logs are systematic records of events generated during the execution of transactions, particularly in interactions involving smart contracts. Each log entry contains *log index*, *data*, and *topics*, crucial to identifying and categorizing specific events such as token transfers, approvals, swaps, minting, and voting.

These logs are emitted using the *emit* keyword within the smart contract code and play a pivotal role in monitoring activities, triggering actions within decentralized applications, and enabling event-driven programming. For instance, Decentralized Exchanges (DEXs) emit events upon trade executions, enabling user interfaces to update displays with current trade information. These logs are stored in transaction receipts, offering a gas-efficient method to capture transient event data without permanently altering the blockchain's state. Among the vast array of data accessible on EVM-based blockchains, transaction logs stand out as crucial sources of information for researchers, developers, and users. They facilitate analyses of various token transfer patterns and support blockchain analysis research, which is the focus of our contributions.

In our dataset, transaction logs should be sorted by the *blockNumber*, *transactionIndex*, and *logIndex* attributes to maintain the correct order in which they are stored on the blockchain. This is particularly important when analyzing the different states of a blockchain before and after the execution of a transaction that triggers a smart contract function.

Topics Attributes. The interpretation of the *topics* attributes ($topics_0$, $topics_1$, $topics_2$, and $topics_3$) depends on the implementation details of the invoked function within a smart contract. Typically, $topics_0$ represents the event name, while subsequent topics represent indexed parameters of the event. The "data" attribute contains non-indexed event parameters. For example, in the context of a token transfer event, $topics_0$ might indicate the event name *Transfer*, $topics_1$ and $topics_2$ could respectively denote sender and receiver addresses, and *data* would typically represent the number of tokens transferred.

Hashing and Signatures. $topics_0$ corresponds to the hashed function signature using *keccak256* [10]. This signature consists of the function name followed by its parameter types. For example, the signature of a typical *Transfer* event is *Transfer(address,address,uint256)*. After hashing it with keccak256, the result becomes *0xddf2* $\cdots$ *b3ef*,[5] which is the $topics_0$. Below is a Python code snippet demonstrating how to verify if a given signature matches $topics_0$:

```
import web3
def check_sig(sig, topics_0):
    return web3.Web3.keccak(text=sig).hex() == topics_0
```

[5] We shortened $topics_0$ to *0xddf252ad* $\cdots$ *f523b3ef* for better visualization in the paper.

Event Mapping. We provide a mapping of the top 90 most frequently invoked events within the ZKsync dataset in our GitHub repository under *./src/utils.py#events_dict*. This mapping facilitates the parsing of the majority of events in our dataset. The mapping is structured as a dictionary where the topics_0 hex value serves as the key, and the corresponding value is a dictionary containing the parsed event name and its function signature. For instance, the Transfer event is represented as follows within the map:

```
events_dict["0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef"] = {
    "name": "Transfer",
    "signature": "Transfer(address,address,uint256)"}
```

2.5 L2 to L1 Logs

L2 to L1 logs are messages emitted by the ZKsync L2 network and transmitted to the Ethereum L1 network. They are essential for maintaining communication between the two layers, ensuring the security and integrity of transactions and data transfers. In ZKsync, the L1 smart contract verifies these communications by checking the messages alongside the ZKP. The only *provable* part of the communication from L2 to L1 is the native L2 to L1 logs emitted by the Virtual Machine (VM). These logs can be generated using the *to_l1 opcode* [7]. We refer the reader to the ZKsync documentation for more details [5].

3 Example Analyses

This section demonstrates some analyses that can be performed using our ZKsync dataset. Each subsection concentrates on a specific topic and utilizes their respective data (e.g., blocks, transactions, receipts, and logs). In this paper, we consider all activity that starts with block 561,367, the first block on April 1st, until block 29,710,983, the last block in the ZKsync dataset.

This section should serve as a starting point for other researchers wishing to use this dataset for their research. The code for generating this analysis is discussed in Table 1 and will be available in a GitHub repository.

3.1 Gas Usage and Transaction Fees

To start, we look into transactions, gas usage, and fees. All results are generated from two main sources of data, namely blocks and transaction receipts. Note that we are using the transaction receipts instead of the transaction data because the receipts are the source of the actual units of gas used and the final transaction fee paid by users.

Figure 2 shows the daily transactions executed in ZKsync Era over the period analyzed. The network processed an average of 905,194 transactions daily, with a significant spike in December 2023 that accounts for 5,362,921 transactions in a single day. This spike was due to a boom in inscriptions, which became very popular for memecoin traders around this time. Messias *et al.* [26] did a comprehensive review of this phenomenon across various rollups. Since the spike stabilized, we have seen a slight increase in daily transactions, which are now hovering around 1.2 million transactions per day.

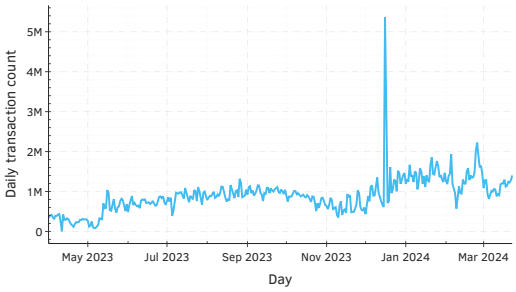
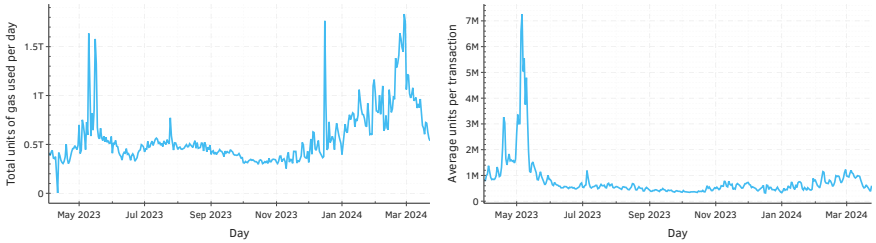


Fig. 2: Total transactions executed per day.



(a) Total gas units. (b) Average gas unit.

Fig. 3: Comparison of gas unit usage over time: (a) Total gas units used per day; and (b) Average gas units used per transaction.

Considering gas usage, Figure 3a illustrates the total number of gas units used each day in the ZKsync Era. On average, the network has processed 567,111,979,658 gas units per day in the last year. However, gas usage has been relatively volatile. Particularly, it has experienced three significant spikes of more than 1.5 trillion gas units per day, which we detail next.

May 2023. 30% of all gas used during the month of May 2023 can be attributed to the *zkApes airdrop*. The address receiving the most gas units was the *SyncSwap router* contract, which is used to execute swaps in the largest DEX in ZKsync Era. During this time, we see more than 10%

December 2023. This was caused by the inscriptions boom we discussed before [26].

March 2024. The gas utilization during this spike is much more distributed among transaction receivers, with the top receivers being DEX routers, such as the *SyncSwap V2 router* and the *Mute.io router* (which recently re-branded to Koi), and token contracts, such as *USDC.e* and *SOUL*. This dispersed distribution of gas usage coupled with the observed steady growth in both gas usage and transaction count up to the spike suggests that it is due to the increased trading activity on ZKsync Era.

We can further contextualize gas utilization by looking at Figure 3b, which shows the average number of gas units used per transaction. We see that in May 2023, transactions were using on average a significantly higher number of gas units (reaching 7,265,323 units per transaction) compared to the average over the 1-year period (784,149 units).

Finally, we can look at the transaction fees. Figure 4 shows the average transaction fees over time. In other words, this is the cost that users have paid, on average, to

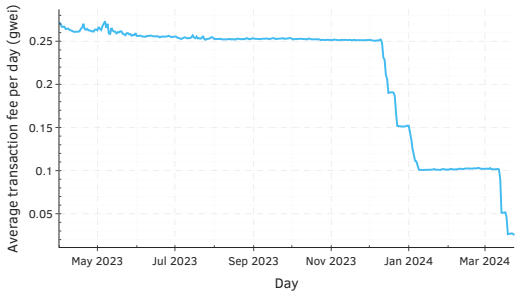


Fig. 4: Average transaction fee in gwei per unit of gas by day.

submit one unit of gas to ZKsync Era. Transaction fees were fairly stable during 2023, with an average of 0.25 gwei per gas unit.

However, in late 2023, a key upgrade led to a significant decrease in ZKsync transaction fees. Concretely, the implementation of the new prover, Boojum [24], marked a significantly reduced hardware requirement to run a Zero-Knowledge (ZK) prover and thus allowed a reduction of transaction fees to 0.1 gwei per unit of gas.

Then, in March 2024, the Dencun upgrade was implemented and deployed on the Ethereum mainnet. This upgrade, brought in EIP-4844, introduced a new type of transaction that can store "blobs" of data in the beacon node for 14 days [44]. Blobs have their independent fee model and submitting data as blobs is much cheaper than the previously used call-data [32]. ZKsync Era was one of the first rollups using blobs to publish the data related to its state changes, thus further reducing transaction fees to 0.025 gwei per unit of gas.

3.2 Events and Contract Deployments

Events are an important source of additional data about activity on EVM chains. They correspond to data emitted by smart contracts and stored on-chain. Next, we analyze this data and report on some specific events. Recall that events can be obtained from the logs data in the ZKsync dataset as discussed in Section 2.

Figure 5 shows the top 15 event types with the most emitted events during the period under analysis. Different contracts may emit the same event type, so what distinguishes them is the event's function signature and, thus, its respective hash.

There is a significant overrepresentation of the top 4 event types, with *Transfer* events being by far the most significant type (70.9% of all events are Transfer events). This is not unexpected as this event is emitted every time an ERC-20 token is transferred between two addresses in ZKsync Era. This occurs in simple token transfers and other standard contract operations such as swaps in DEXs.

Before looking at *Transfer* events in more detail, we should highlight a particularity in how ZKsync Era implements transaction fee collection. In this chain, every transaction generates two additional ETH transfers — one for the initial payment of the transaction fees from the transaction submitter and another with a transaction fee refund (after all L1 and proving costs are accounted for). These transaction fee transfers always appear as ETH transfers from or to the address *0x8001*[6] and thus generate *Transfer* events. These specific events account for 38.2% of all events emitted, and if we

[6] We shortened this address for better visualization in the paper.

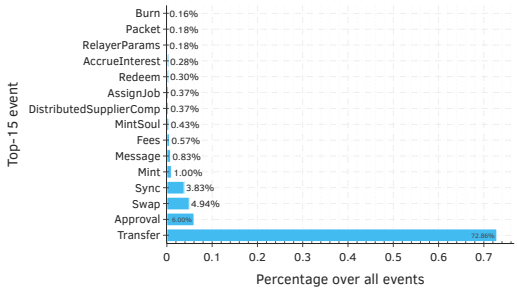
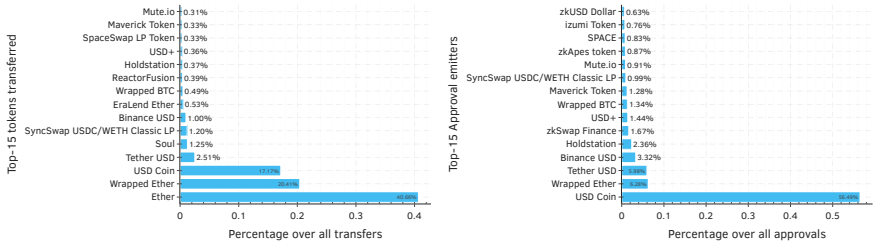


Fig. 5: Top 15 event types with the highest number of events emitted between April 1st, 2023 and March 24th, 2024, and their percentage over all events emitted. Percentages *include* ETH transfers generated from transaction fee payments.



(a) Top 15 ERC-20 tokens with the highest number of *Transfer* events emitted.

(b) Top 15 contracts with the highest number of *Approval* events emitted.

Fig. 6: Percentage of the top 15 contracts with the highest Transfer and Approval events emitted between April 1st, 2023 and March 24th, 2024: (a) Transfer events, where percentages *exclude* ETH transfers generated from transaction fee payments; and (b) Approval events.

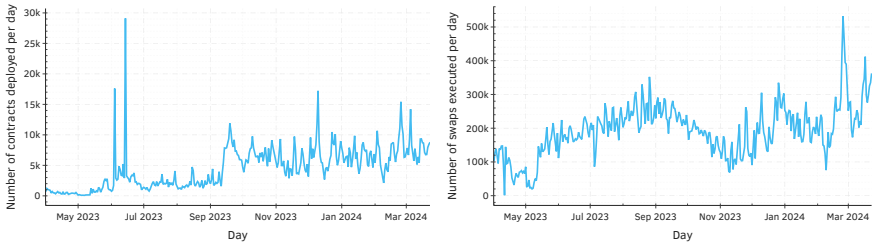
exclude them entirely from the event dataset, Transfer events only account for 56.1% of these filtered events.

After filtering the Transfer events generated by fee management ETH transfers, we can explore the top tokens transferred by examining the contract address that emitted the event. Figure 6a displays the 15 tokens most frequently transferred during the period. Native ETH, Wrapped ETH, and Bridged USDC (USDC.e) are the most transfers, accounting collectively for 78.4%. We can also see some other stablecoins, such as Tether USD or Binance USD, and Liquidity Provider (LP) tokens associated with DEXs on ZKsync Era, such as SyncSwap and SpaceSwap.

Approvals are the second-largest event type, with 6% of events emitted. Similarly to the Transfer events, we can see which contracts emit these events. Figure 6b shows these top 15 emitters.

The top emitters are also ERC-20 token contracts, and these events represent an owner "approving" a spender to transfer a predefined amount of tokens they hold. This is common in bridged assets, for example, as is the case of Bridged USDC (USDC.e) and Wrapped ETH, the top 2 emitters.

Swaps and *Syncs* are the third and fourth most frequently emitted event types, accounting for 4.9% and 3.8% of all events emitted, respectively. These events are a key source of data for analyzing DEXs. However, we will explore these more in-depth in the next subsection.



(a) Number of contracts deployed. (b) Number of swap events emitted.

Fig. 7: Daily total number of contracts deployed (in a) and daily number of swap events emitted (in b) on ZKsync Era.

Finally, we examine contract deployments. Every time a contract is deployed, a specific event type is emitted, which allows us to easily track this metric. Although this event type is not among the 15 most emitted event types, it is still an important metric for network activity. In ZKsync, the contract deployment event is named *ContractDeployed*.

Figure 7a shows the number of contracts deployed on ZKsync each day. Before September 2023, developers averaged 2510 contract deployments per day. However, there were two major spikes during this time, the first reaching 29,114 and the second reaching 17,602 contract deployments in a single day. Then, after September 2023, contract deployments increased to a daily average of 6672.

3.3 Swaps

After a high-level look at these events, we now focus on a specific event relevant to understanding activity on DEXs — the *Swap* event. Recall that swap events account for 4.9% of all events emitted on ZKsync.

Swap events are emitted every time a successful swap is performed in a DEX. These events contain information about the contract that emits the event, the amounts of each token being traded, and the wallets involved in the trade.

Figure 7b shows the number of swap events emitted each day during the analyzed period, which is equivalent to the number of swaps performed each day on ZKsync DEXs. We have a long-term trend of increasing the number of daily swaps, with a peak around March 2024 (which reached 531,819 swaps in a single day). During this year, users performed an average of 192,009 swaps per day.

We also see which LP contract was involved in the swap by looking at the contract address that emitted the event. Depending on the DEXs protocol, this may be the actual liquidity pool involved in the swap (e.g., SyncSwap and Koi) or a generic LP contract managing all the pools in the DEX (e.g., SpaceSwap and PancakeSwap). To understand which tokens were being traded in these generic LP contracts, we would have to process the transfer events emitted in the same transaction of these Swap events.

Figure 8 shows the top 15 contracts that emitted the most swaps even during the period under analysis. The largest emitter is, by a significant margin, the USDC/WETH pool on SyncSwap, which accounts for almost a third of all emitted Swap events. All combined swap events from SpaceSwap represent 11% of all swaps, followed by two USDC/WETH pools from other DEX.

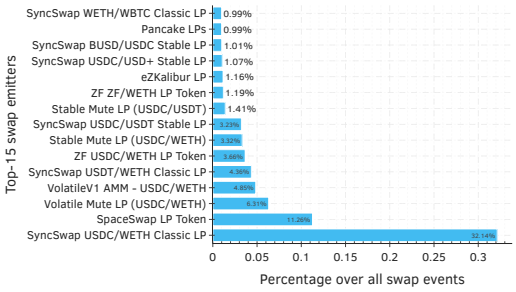


Fig. 8: Top 15 contracts emitting the most Swap events between April 1st, 2023 and March 24th, 2024, and their percentage over all Swap events emitted.

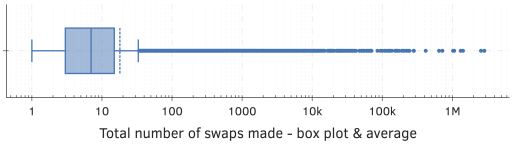


Fig. 9: Distribution of swaps per unique wallet addresses.

Finally, we can use the "receivers" field to analyze swappers. In other words, this field provides the wallet address receiving the swapped token. Figure 9 displays the distribution of swaps made by unique wallet addresses.

We note that most users performed less than 50 swaps during the period under analysis. Concretely, the percentile 95% of this distribution is 42 swap events. However, the distribution has a significant skew to the right, with a few addresses generating a large number of swaps in the year examined. These large "traders" are usually routers and other protocols that interact with DEXs; thus, they represent many end-users. Examples include the top swappers, which are the *Mute.io router*, the *SpaceFi router*, and the *Odos V2 router*, respectively.

4 Future Directions

Among the data analysis presented in this paper, our ZKsync dataset can be applied to different studies. We list some of them below, where this dataset can be valuable.

MEV and Arbitrage. MEV and arbitrage have been extensively studied in L1 blockchains [27,39,40,42,45]. However, only recently have new studies shifted their focus to L2s [9,17,18,43], including ZKsync Era. Our dataset can contribute to this type of research by enabling further analysis of MEV on ZKsync Era. For example, one type of MEV known as *backrunning* involves arbitrageurs ensuring their transactions are included immediately after a target transaction. This can be particularly useful in scenarios like liquidation-MEV, where arbitrageurs take advantage of opportunities that arise right after an oracle update [27,39,45]. Another form of arbitrage worth analyzing is CEX-DEX MEV, where arbitrageurs exploit price deviations between Centralized Exchange (CEX) and DEX platforms [21]. Additionally, studying cross-rollup MEV could provide insights into opportunities where arbitrageurs benefit from differences across two or more rollups [17].

Analysis of user activity on chain. Analyzing user activity in the ZKsync chain provides valuable insights into user behavior. This includes examining the transactions

issued by users and their interactions with smart contracts. Such an analysis can also help identify airdrop farmers who create multiple accounts, known as Sybils, although this task can be challenging [15,29]. Another important area of analysis is measuring the impact of users in social media networks driving activities on the blockchain. For example, examining the recent boom in inscriptions in L2s chains can reveal how social networks can influence blockchain activity [26,31]. Additionally, since our dataset includes all event logs, it allows for the analysis of decentralized governance of protocols deployed on ZKsync Era. This involves studying proposals to amend smart contracts and their voting processes. By filtering data related to governance contracts, such as how each user voted and the distribution of these governance tokens among users, it can shed light on the implications of token concentration on decentralized governance, raising concerns about fairness [28,41].

Data Science Analytics. This dataset can also be valuable for users or non-researchers interested in exploring blockchain data. It offers an opportunity for those who wish to better understand blockchains. For example, data scientists can utilize this dataset to explore and analyze blockchain data on public platforms such as Kaggle. Data scientists widely use it to demonstrate their general skills in data analytics, machine learning, and data science. Therefore, this dataset can help data scientists acquire new skills, giving them a competitive edge in the market or improving their prospects of securing a job in a blockchain company.

We hope this dataset proves to be a valuable resource for research groups interested in conducting studies on L2s blockchains and ZKsync Era. We plan to publish our dataset and code in a GitHub repository to facilitate this.

5 Conclusion

At a high level, this paper addresses a critical concern within the scientific community: *the availability and accessibility of blockchain data*. Data-driven research is dependent on high-quality datasets to make meaningful findings. Although blockchain data are theoretically publicly available, practical challenges often prevent researchers, especially those without technical expertise, from obtaining and preprocessing them. In addition, the cost of using external services or deploying infrastructure to run archive nodes can be prohibitive.

To address these challenges, we have collected, pre-processed, and made available our ZKsync dataset to facilitate access for researchers. We also provide a detailed background on blockchains, including blocks, transactions, receipts, and logs, designed to help non-experts understand and utilize the data effectively. To illustrate the potential of this dataset, we offer example analyses and discuss future research directions.

We believe our contribution will be valuable to researchers studying blockchain L2 ecosystems, particularly those interested in ZKsync. This dataset also adds to the existing body of L2 research. It is designed for ease of use, using the Python library Polars to allow straightforward data processing on a local laptop.

Finally, to promote scientific reproducibility and to support further research, we have made our dataset publicly available on GitHub [4].

References

1. EIP-712: Typed structured data hashing and signing. https://eips.ethereum.org/EIPS/eip-712 (2017)

2. EIP-1559: Fee market change for ETH 1.0 chain. https://eips.ethereum.org/EIPS/eip-1559 (2019)
3. EIP-2930: Optional access lists. https://eips.ethereum.org/EIPS/eip-2930 (2020)
4. Data sets and scripts used to analyze the ZKsync Era blockchain. https://github.com/matter-labs/zksync-data-dump (2024)
5. L1 <-> L2 Communication. https://docs.zksync.io/zk-stack/concepts/l1_l2_communication (2024)
6. Reth Book. https://reth.rs (2024)
7. System contracts/bootloader description (VM v1.4.0). https://github.com/code-423n4/2023-10-zksync/blob/main/docs/SmartcontractSection/Systemcontractsbootloaderdescription.md (2024)
8. Transaction Lifecycle. https://docs.zksync.io/zk-stack/concepts/transaction-lifecycle#transaction-types (2024)
9. Bagourd, A., Francois, L.G.: Quantifying mev on layer 2 networks. arXiv preprint arXiv:2309.00629 (2023)
10. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Keccak sponge function family main document. Submission to NIST (Round 2) (2009)
11. bitcoin.org: Bitcoin Core. https://bitcoin.org/en/bitcoin-core (2024)
12. bnbchain.org: BNB Smart Chain. https://www.bnbchain.org/en/bnb-smart-chain (2024)
13. Buterin, V.: A rollup-centric ethereum roadmap (Oct 2020), https://ethereum-magicians.org/t/a-rollup-centric-ethereum-roadmap/4698
14. Day, A., Medvedev, E., AK, N., Price, W.: Introducing six new cryptocurrencies in bigquery public datasets—and how to analyze them. Google Cloud (2019)
15. Fan, S., Min, T., Wu, X., Cai, W.: Altruistic and profit-oriented: Making sense of roles in web3 community from airdrop perspective. In: Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems. CHI '23 (2023)
16. Gogol, K., Fritsch, R., Messias, J., Schlosser, M., Kraner, B., Tessone, C.: Liquid staking tokens in automated market makers (2024)
17. Gogol, K., Messias, J., Miori, D., Tessone, C., Livshits, B.: Cross-rollup mev: Non-atomic arbitrage across l2 blockchains (2024)
18. Gogol, K., Messias, J., Miori, D., Tessone, C., Livshits, B.: Quantifying arbitrage in automated market makers: An empirical study of ethereum zk rollups. arXiv preprint arXiv:2403.16083 (2024)
19. Gogol, K., Messias, J., Schlosser, M., Kraner, B., Tessone, C.: Cross-border exchange of cbdcs using layer-2 blockchain. arXiv preprint arXiv:2312.16193 (2023)
20. Heimbach, L., Kiffer, L., Ferreira Torres, C., Wattenhofer, R.: Ethereum's proposer-builder separation: Promises and realities. In: Proceedings of the 2023 ACM on Internet Measurement Conference. IMC '23 (2023)
21. Heimbach, L., Pahari, V., Schertenleib, E.: Non-Atomic Arbitrage in Decentralized Finance. In: 2024 IEEE Symposium on Security and Privacy (SP) (2024)
22. Johnsson, T.: Efficient compilation of lazy evaluation. In: Proceedings of the 1984 SIGPLAN symposium on Compiler construction (1984)
23. L2Beat: L2Beat: The state of the layer two ecosystem. https://l2beat.com/scaling/summary?sort-by=total&sort-order=desc#layer2s (2024)
24. Matter Labs: Boojum Upgrade: zkSync Era's New High-performance Proof System for Radical Decentralization. https://zksync.mirror.xyz/HJ2Pj45EJkRdt5Pau-ZXwkV2ctPx8qFL19STM5jdYhc (2024)

25. Messias, J., Alzayat, M., Chandrasekaran, B., Gummadi, K.P., Loiseau, P., Mislove, A.: Selfish & opaque transaction ordering in the bitcoin blockchain: The case for chain neutrality. In: Proceedings of the 21st ACM Internet Measurement Conference. IMC '21 (2021)
26. Messias, J., Gogol, K., Silva, M.I., Livshits, B.: The writing is on the wall: Analyzing the boom of inscriptions and its impact on evm-compatible blockchains. arXiv preprint arXiv:2405.15288 (2024)
27. Messias, J., Pahari, V., Chandrasekaran, B., Gummadi, K.P., Loiseau, P.: Dissecting Bitcoin and Ethereum Transactions: On the Lack of Transaction Contention and Prioritization Transparency in Blockchains. In: Proceedings of the Financial Cryptography and Data Security (FC'23) (May 2023)
28. Messias, J., Pahari, V., Chandrasekaran, B., Gummadi, K.P., Loiseau, P.: Understanding blockchain governance: Analyzing decentralized voting to amend defi smart contracts (2024)
29. Messias, J., Yaish, A., Livshits, B.: Airdrops: Giving money away is harder than it seems. arXiv preprint arXiv:2312.02752 (2023)
30. Offchain Labs: A gentle introduction to Arbitrum. https://docs.arbitrum.io/intro (2024)
31. Omena, J.J., Messias, J., Gouveia, F., Ventura, R.: Digital methods for blockchain research. https://www.researchgate.net/publication/382511569_Digital_Methods_for_Blockchain_Research (2024)
32. Omkar Godbole: Layer 2 Blockchains Become Cheaper After Ethereum's Dencun Upgrade. https://www.coindesk.com/markets/2024/03/14/layer-2-blockchains-become-cheaper-after-ethereums-dencun-upgrade (2024)
33. Optimism Foundation: Optimism. https://www.optimism.io (2024)
34. Pandas: Pandas: Python data analysis library. https://pandas.pydata.org (2024)
35. Paradigm: Paradigm Data Portal. https://www.paradigm.xyz/oss/portal (2024)
36. Paradigm: Paradigm Data Portal. https://github.com/paradigmxyz/paradigm-data-portal (2024)
37. Polars: Lazy / Eager API: Polars user guide. https://docs.pola.rs/user-guide/concepts/lazy-vs-eager (2024)
38. Polars: Polars: Dataframes for the new era. https://pola.rs (2024)
39. Qin, K., Zhou, L., Gervais, A.: Quantifying blockchain extractable value: How dark is the forest? In: 2022 IEEE Symposium on Security and Privacy (SP) (2022)
40. Qin, K., Zhou, L., Livshits, B., Gervais, A.: Attacking the defi ecosystem with flash loans for fun and profit. In: International conference on financial cryptography and data security (2021)
41. Sharma, T., Potter, Y., Pongmala, K., Wang, H., Miller, A., Song, D., Wang, Y.: Unpacking how decentralized autonomous organizations (daos) work in practice. In: 2024 IEEE International Conference on Blockchain and Cryptocurrency (ICBC) (2024)
42. Torres, C.F., Camino, R., State, R.: Frontrunner jones and the raiders of the dark forest: An empirical study of frontrunning on the ethereum blockchain. In: 30th USENIX Security Symposium (USENIX Security 21) (2021)
43. Torres, C.F., Mamuti, A., Weintraub, B., Nita-Rotaru, C., Shinde, S.: Rolling in the shadows: Analyzing the extraction of mev across layer-2 rollups. arXiv preprint arXiv:2405.00138 (2024)
44. Vitalik Buterin and Dankrad Feist and Diederik Loerakker and George Kadianakis and Matt Garnett and Mofi Taiwo and Ansgar Dietrichs: EIP-4844:

Shard Blob Transactions. https://github.com/ethereum/EIPs/blob/master/EIPS/eip-4844.md (2022)
45. Weintraub, B., Torres, C.F., Nita-Rotaru, C., State, R.: A flash (bot) in the pan: measuring maximal extractable value in private pools. In: Proceedings of the 22nd ACM Internet Measurement Conference (2022)

A Additional Information for Data Schemas

In this section, we present the attributes available in our dataset, including their data types and brief descriptions. Our dataset consists of one year of ZKsync data, containing all blocks, transactions, receipts, and logs included in the ZKsync blockchain.

Table 2 describes the attributes of each block added to the ZKsync blockchain. Table 3 details the transaction data, including their data types and descriptions. Table 4 provides information on transaction receipts, which are generated immediately after a transaction is executed and added to a block. This is useful for computing the actual gas spent on a transaction and the corresponding transaction fees paid by users.

Next, Table 5 contains information about transaction logs, which is crucial for analyzing changes in smart contract states such as account balances, token transfers, votes cast, and swaps. Finally, Table 6 presents the attributes related to the messages emitted by ZKsync to the Ethereum mainnet.

B Glossary

Following is a list of important notations used in this paper.

Acronyms

AMM Automated Market Maker
CEX Centralized Exchange
DEX Decentralized Exchange
EVM Ethereum Virtual Machine
L1 Layer-1 blockchain
L2 Layer-2 blockchain
LP Liquidity Provider
MEV Maximal-Extractable Value
RPC Remote Procedure Call
TVL Total Value Locked
VM Virtual Machine
ZK Zero-Knowledge
ZKP Zero-Knowledge Proof

Attribute	***Type***	***Description***
hash	str	Unique identifier for the block.
parentHash	str	Unique identifier of the parent block.
sha3Uncles	str	SHA-3 hash of the uncles' block headers. On ZKsync it is set to $0x1dcc \cdots 9347$ since there are no uncle blocks.
miner	str	Address of the miner who mined the block. This is set as Null address ($0x0$) in all ZKsync blocks since it does not have miners or block validators.
stateRoot	str	Root hash of the state trie. Set to Null address ($0x0$).
transactionsRoot	str	Root hash of the transaction trie. Set to Null address ($0x0$).
receiptsRoot	str	Root hash of the receipts trie. Set to Null address ($0x0$).
number	i64	Block number or height.
l1BatchNumber	str	L1 batch number, related to the sequence of batches submitted to Layer 1 on zkRollup systems.
gasUsed	i64	Total amount of gas used by all transactions in the block.
gasLimit	i64	Maximum amount of gas that can be used by all transactions in the block. Set to 4,294,967,296 (2^{32}) units of gas.
baseFeePerGas	i64	Base fee, in Wei ($10^{-18}ETH$), per unit of gas.
extraData	str	Extra data included by the miner in the block. Set to $0x$ since there are no miners.
logsBloom	str	Bloom filter for the logs of the block. Set to $0x0 \cdots 0$.
timestamp	i64	Unix timestamp of when the block was collated.
l1BatchTimestamp	str	L1 batch timestamp (in HEX format) associated with the block.
difficulty	i64	Difficulty target for mining the block. Set to 0 since there is no mining.
totalDifficulty	i64	Cumulative difficulty of the blockchain up to and including this block. Set to 0 since there is no mining.
sealFields	list[null]	Seal fields containing proof-of-work or proof-of-stake information. List containing null as value.
uncles	list[null]	Uncle blocks that were mined but not included in the main chain. List with Null value since there is no mining.
size	i64	Size of the block in bytes. Set to 0.
mixHash	str	Hash used in the mining process to prove that enough computational work has been performed. Set to $0x0 \cdots 0$ since there is no mining.
nonce	str	Value used in the mining process to find a valid block hash. Set to $0x0 \cdots 0$ since there is no mining.

Table 2: Description of block attributes on ZKsync.

Attribute	***Type***	***Description***
blockHash	str	Unique identifier of the block containing the transaction.
blockNumber	i64	Block number or height containing the transaction.
chainId	i64	Identifier of the blockchain network. Set to 324 that represents ZKsync chain.
from	str	Address of the sender of the transaction.
gas	i64	Amount of gas provided as *gasLimit* for the transaction.
gasPrice	i64	Price per unit of gas the sender is willing to pay.
hash	str	Unique identifier for the transaction.
input	str	Data sent along with the transaction. In HEX code.
l1BatchNumber	str	L1 batch number related to the transaction in zkRollup systems.
l1BatchTxIndex	str	Index of the transaction in the L1 batch.
maxFeePerGas	i64	Maximum fee, in Wei (10^{-18} ETH), per unit of gas.
maxPriorityFeePerGas	i64	Maximum priority fee, in Wei (10^{-18} ETH), per unit of gas.
nonce	i64	Number of transactions sent by the sender prior to this one.
r	str	First part of the ECDSA signature.
s	str	Second part of the ECDSA signature.
to	str	Address of the receiver of the transaction.
transactionIndex	i64	Index of the transaction within the block.
type	i64	Type of transaction divided into 5 categories: *Legacy* (0 or *0x0*), *EIP-2930* (1 or *0x1*) [3], *EIP-1559* (2 or *0x2*) [2], *EIP-712* (113 or *0x71*) [1], and *Priority* (255 or *0xff*). See details in ZKsync documentation [8].
v	f64	Recovery id of the ECDSA signature.
value	str	Amount of tokens (in Wei and in HEX format) that are transferred in the transaction to the recipient address (*to*).

Table 3: Description of transaction attributes on ZKsync.

Attribute	**Type**	**Description**
blockHash	str	Unique identifier of the block containing the transaction.
blockNumber	i64	Block number or height containing the transaction.
contractAddress	str	Address of the contract created by the transaction, if applicable.
cumulativeGasUsed	i64	Total amount of gas used when the transaction was executed in the block. Set to 0 on ZKsync.
effectiveGasPrice	i64	Actual price per unit of gas, in Wei (10^{-18} ETH), paid.
from	str	Address of the sender of the transaction.
gasUsed	i64	Amount of gas used by the transaction.
l1BatchNumber	str	L1 batch number related to the transaction in zkRollup systems.
l1BatchTxIndex	str	Index of the transaction in the L1 batch.
logsBloom	str	Bloom filter for the logs of the transaction. Set to *0x0* $\cdots$ *0* on ZKsync.
root	str	State root after the transaction is executed.
status	i64	Status of the transaction (1 for success, 0 for failure).
to	str	Address of the receiver of the transaction.
transactionHash	str	Unique identifier for the transaction.
transactionIndex	i64	Index of the transaction within the block.
type	i64	Type of transaction divided into 5 categories. Refer to Table 3 and to ZKsync documentation [8] for details.

Table 4: Description of transaction receipt attributes on ZKsync.

Attribute	Type	Description
address	str	Address of the contract that generated the log.
blockHash	str	Unique identifier of the block containing the transaction.
blockNumber	i64	Block number or height containing the transaction.
data	str	Data contained in the log. This can be used, for example, to extract the amount of tokens transferred from one user to another.
l1BatchNumber	str	L1 batch number related to the log in zkRollup systems.
logIndex	i64	Index of the log within the block.
logType	null	Type of log. Set to *null* on ZKsync.
removed	bool	Indicates whether the log was removed (*true*) or not (*false*).
transactionHash	str	Unique identifier for the transaction.
transactionIndex	i64	Index of the transaction within the block.
transactionLogIndex	str	Index of the log within the transaction. In HEX format.
topics_0	str	First topic of the log. This is typically referred to as the name of the event encoded in hexadecimal (HEX) format.
topics_1	str	Second topic of the log.
topics_2	str	Third topic of the log.
topics_3	str	Fourth topic of the log.

Table 5: Description of transaction log attributes on ZKsync.

Attribute	Type	Description
blockHash	str	Unique identifier of the block containing the log.
blockNumber	str	Block number or height containing the log.
isService	bool	Indicates whether the log is a service log (*true*) or not (*false*).
key	str	Key associated with the log that could be used to carry some data with the log.
l1BatchNumber	str	L1 batch number related to the log in zkRollup systems.
logIndex	str	Index of the log within the block.
sender	str	It is the value of *this* in the frame where the L2→L1 log was emitted.
shardId	str	It is the id of the shard the opcode was called. It is currently set to 0.
transactionHash	str	Unique identifier for the transaction associated with the log.
transactionIndex	str	Index of the transaction within the block.
transactionLogIndex	str	Index of the log within the transaction.
txIndexInL1Batch	str	Index of the transaction within the L1 batch.
value	str	Value associated with the log that could be used to carry some data with the log.

Table 6: Description of L2 to L1 log attributes on ZKsync.