✅

zkSync

Executive Summary
Introduction
Goals & Methodology
Results & Discussion
Transaction Pricing Mechanism Specifics
Basic Differences:
L2 fees
Refunds
Data Analysis
Fee Revenue
Costs
Profitability
Conclusion
Data Tables

Executive Summary

The following report examines Transaction Fees on zkSync Era and its profitability, focusing on both Revenue (gas fees earned) and Costs (transaction fee payments for Data Availability and Proof submission on Ethereum by ZK Sync), excluding proof generation costs as they are not publicly available.

Transactions on zkSync are impacted by L1 gas costs needed to publish public data for batches and verify proofs. To manage this, zkSync-specific EIP712 transactions include a gas_per_pubdata_limit field, which sets the maximum gas price the operator can charge users per byte of pubdata. This helps mitigate the impact of fluctuating gas prices on users.

Unlike Ethereum, where the intrinsic transaction cost (21000 gas) covers updating user balances, nonce, and signature verification, zkSync does not include these prices in the intrinsic transaction costs. Also, refunds can be issued for unused system resources and overpaid computation. This is necessary due to the relatively large upfront payments required in zkSync to ensure DDoS protection. Total profit up to may 20th: 13991.90 ETH.

Impact on our design: Implementing a similar mechanism to manage L1 gas cost volatility can protect users from unpredictable fee fluctuations. Redefining our intrinsic cost structure to exclude certain intrinsic costs can optimize fee allocation, leading to more efficient resource usage and potentially lower transaction fees. Implementing a refund mechanism in the transaction fee formula can enhance user experience and free up storage slots.

Introduction

ZkSync Era is a ZK rollup we are all pretty familiar with. All transactions are proven on Ethereum, users enjoy the same security level as in Ethereum. It uses a centralized sequencer. Transactions can also be submitted via L1. Native support of ECDSA signatures: no special operation is required to register the user’s private key. Any account can be managed in L2 with the same private key that is used for L1. L1 to L2 smart contract messaging is possible. Transaction fees can be paid with ERC20 tokens (e.g. USDC) thanks to native account abstraction and paymasters.

Goals & Methodology

The main goal of the report is to analyse zkSync Era data before and after EIP4844 upgrade and to determine:

- Average cost of verification
- Number of transactions over time
- Data Availability cost (ETH)

- Average time between submissions for both Data Availability and ZK Proofs
- Fee (movement, adj fee)
- Total revenue
- Economics specifics

This research is performed by obtaining both Ethereum Mainnet and zkSync data using Dune Analytics and performing statistical analysis. The Pricing mechanism and other specifics are acquired by reading official documentation, blogs, and other materials on zkSync Era network.

Results & Discussion

Transaction Pricing Mechanism Specifics

As with most Zk rollups it includes additional costs for publishing to L1 and for proof generation. Transaction prices mostly depend on L1 gas prices.

Basic Differences:

Lets follow the documentation cover the fee model differences in a bit more detailed way:

- **Storage**
  - On Ethereum, accessing a storage slot or account for the first time incurs a gas charge. ZKsync uses a similar mechanism but with differences. It supports "warm" and "cold" storage slots, where users are initially charged the maximum (cold) cost and then refunded if the slot is "warm." This ensures users always have enough gas for the worst case. The refund process is managed by the operator.
- **Code Decommitment and Account Access Costs**
  - ZKsync separates account balances, nonces, and bytecodes in storage. Transforming a code hash into its preimage is called code decommitment. When a contract with a certain code hash is called, if it's the first time this bytecode has been decommitted, the user is charged the full cost. Otherwise, the user does not pay for decommitment. This process is partially enforced by circuits, ensuring correctness as long as enough gas is available.
- **Memory Pricing**
  - In ZKsync, user contracts receive 2^12 bytes of free memory initially, and kernel space (system) contracts receive 2^21 bytes before charging users linearly based on memory length. Unlike Ethereum, ZKsync does not use a quadratic component for memory expansion pricing.
- **Different Intrinsic Costs**
  - On Ethereum, the intrinsic transaction cost (21000 gas) covers updating balances, nonce, and signature verification. In ZKsync, these costs are not included in intrinsic transaction fees due to native support for account abstraction. Each account type may have its own transaction cost, potentially enabling cheaper transactions through zk-friendly signature schemes or optimizations. ZKsync transactions have small intrinsic costs for bootloader processing, measured through testing and hardcoded.
- **Transactions are processed on a first-come, first-served basis**

L2 fees

To process the batch ZKsync has to pay for proving, generating and committing to it. That means that the batch overhead consists of:

- L2 gas (proving the circuits)
- L1 gas (proof verification, general processing)

Essentially transaction pays for the batch overhead proportionally to how close the transaction brings the batch to being *sealed (prepared for proof verification and submission on L1)*

The batch has limited resources:

- Time - it should not take too much time for batch to be closed. They essentially use batch gas limit to ensure this. (80 million)
- Number of transactions - it cannot take more than certain number of transactions per batch (10000)
- The memory of the bootloader - it needs to store the transaction's ABI encoding in its memory & this fills it up
- Pubdata bytes - most nodes have a limit of 128kb per transaction when submitting to L1 so batches have this limit as well

Each transaction spends the batch overhead proportionally to how closely it consumes these resources. Because they cannot know exactly they charge for the worst case and refund at the end of transaction. The recommended maximum gas that a transaction can spend on computation is defined by MAX_TRANSACTION_GAS_LIMIT to protect against DDoS attacks.

Now lets see how gas prices are calculated:

At the start of each batch, the operator provides the following two parameters:

- FAIR_L2_GAS_PRICE- minimal L2 gas price that the operator is willing to accept. It is expected to cover the cost of proving/executing a single unit of zkEVM gas, the potential contribution of usage of a single gas towards sealing the batch, as well as congestion.
- FAIR_PUBDATA_PRICE - which is the price of a single pubdata byte in Wei. Similar to the variable above, it is expected to cover the cost of publishing a single byte as well as the potential contribution of usage of a single pubdata byte towards sealing the batch.

Essentially the gas price will be one of the above mentioned parameters, depends on which one is higher.

$$\text{baseFee} := \max\left(\text{fairL2GasPrice}, \left\lceil \frac{\text{fairPubdataPrice}}{\text{MAX\_L2\_GAS\_PER\_PUBDATA()}} \right\rceil\right)$$

$$\text{gasPerPubdata} := \left\lceil \frac{\text{pubdataPrice}}{\text{baseFee}} \right\rceil$$

MAX_L2_GAS_PER_PUBDATA = 2^20

Calculation of FAIR_L2_GAS_PRICE and FAIR_PUBDATA_PRICE is as follows:

- **Constants**:
  - BATCH_OVERHEAD_L1_GAS: L1 gas overhead for a batch.
  - COMPUTE_OVERHEAD_PART: Represents the likelihood of batch sealing due to computational resources (range 0 to 1).
  - MAX_GAS_PER_BATCH: Maximum gas a batch can use.
  - PUBDATA_OVERHEAD_PART: Represents the likelihood of batch sealing due to pubdata (range 0 to 1).
  - MAX_PUBDATA_PER_BATCH: Maximum pubdata a batch can use.
- **Fluctuating Variables**:
  - MINIMAL_L2_GAS_PRICE: Minimum acceptable L2 gas price, which is the price that should cover the proving and additional premium for congestion.

- PUBDATA_BYTE_ETH_PRICE: Minimum acceptable price in ETH per calldata or blob byte.

Fair L2 gas price is calculated as follows:

$$\text{fair\_L2\_GasPrice} = \text{MINIMAL\_L2\_GAS\_PRICE} + \frac{\text{COMPUTE\_OVERHEAD\_PART} \times \text{BATCH\_OVERHEA}}{\text{MAX\_GAS\_PER\_BATCH}}$$

Fair pubdata price is calculated as follows:

$$\text{fairPubdataPrice} = \text{PUBDATA\_BYTE\_ETH\_PRICE} + \frac{\text{PUBDATA\_OVERHEAD\_PART} \times \text{BATCH\_OVER}}{\text{MAX\_PUBDATA\_PER\_BATCH}}$$

Constants for transaction slot and memory are as follows;

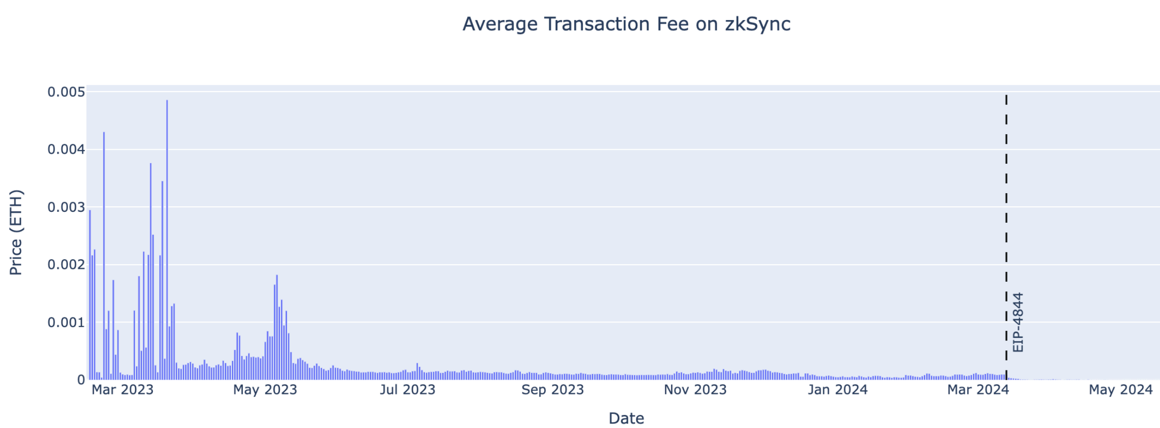
- TX_OVERHEAD_GAS: 10000 gas for including a transaction in a batch.
- TX_MEMORY_OVERHEAD_GAS: 10 gas per byte of bootloader memory used.

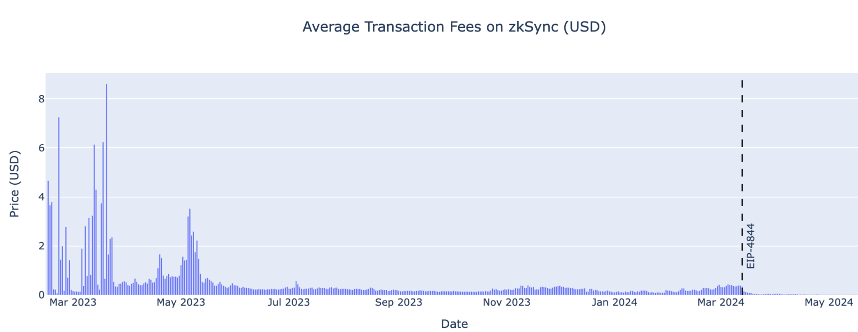
Refunds

Refunds for the batch overhead are probabilistic, covering operator expenses over time. ZKsync Era manages refunds for repeated writes by optimizing pubdata for storage changes, with the operator enforcing these refunds.

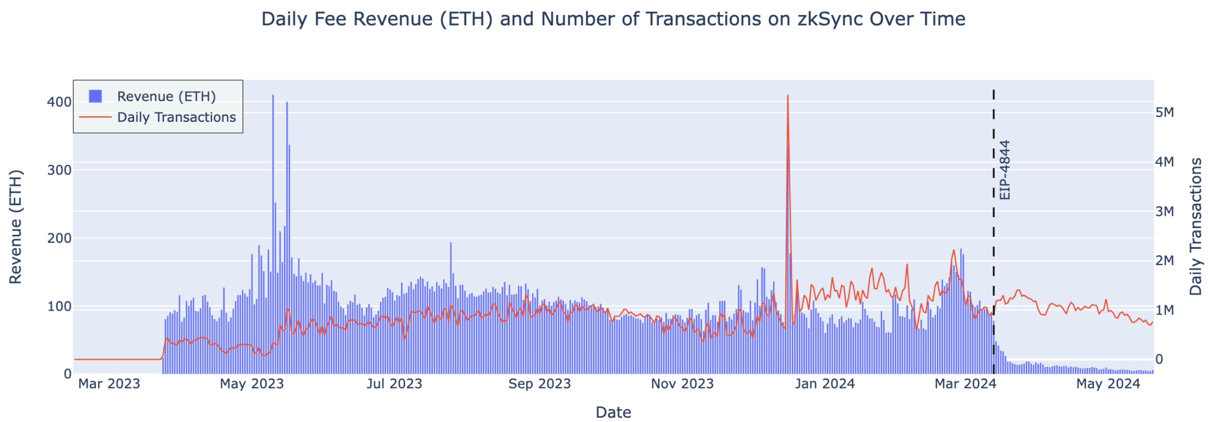
Data Analysis

Fee Revenue



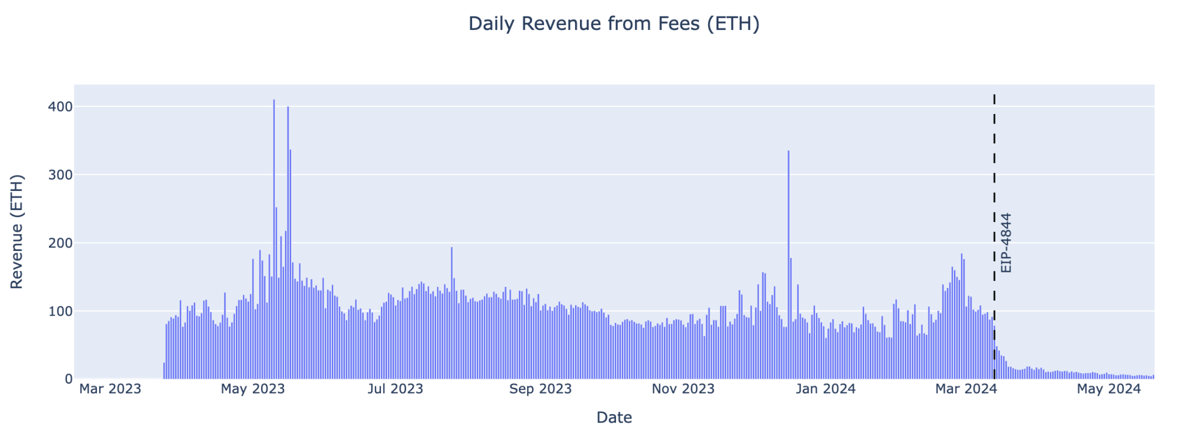


We can see that the average transaction fee dropped substantially after enabling blobs.



We can see that the number of daily transactions on average around 1m. We see no increase in number of transactions after enabling blobs so we can assume that the demand is inelastic with stochastic variations. We will take a look at the EIP-4844 impact on profits in the profitability section.

.



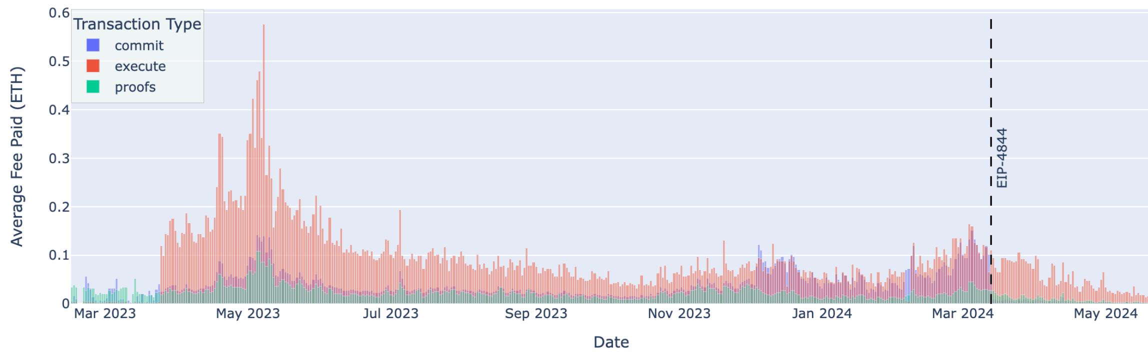
Costs

commit - DA/Blobs

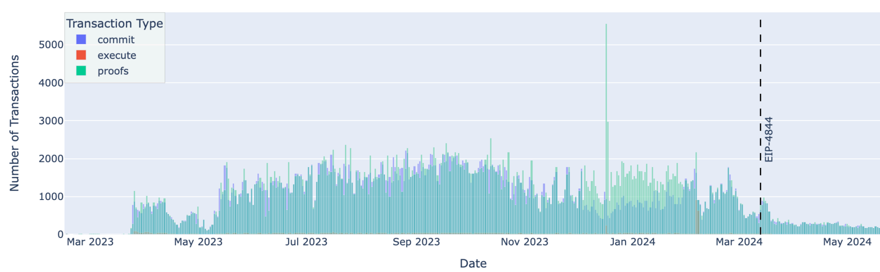
execute - transaction execution

proofs - proofs

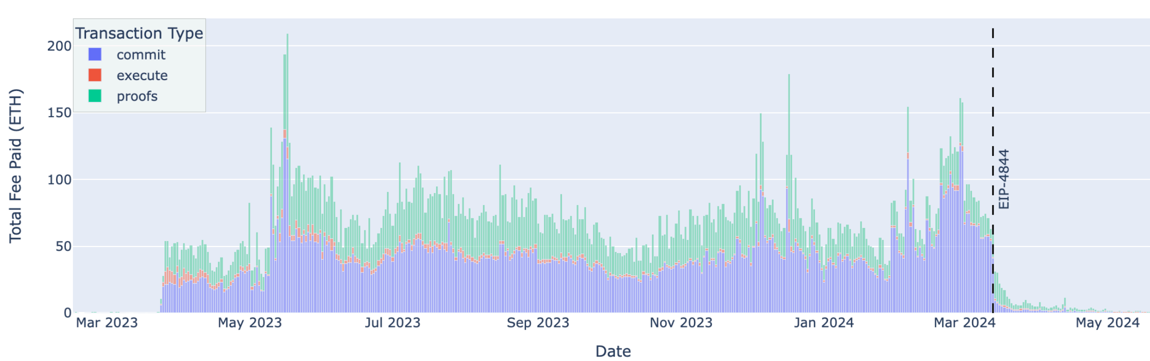
Average Fee Paid (per transaction) in ETH for execute, commit and proofs

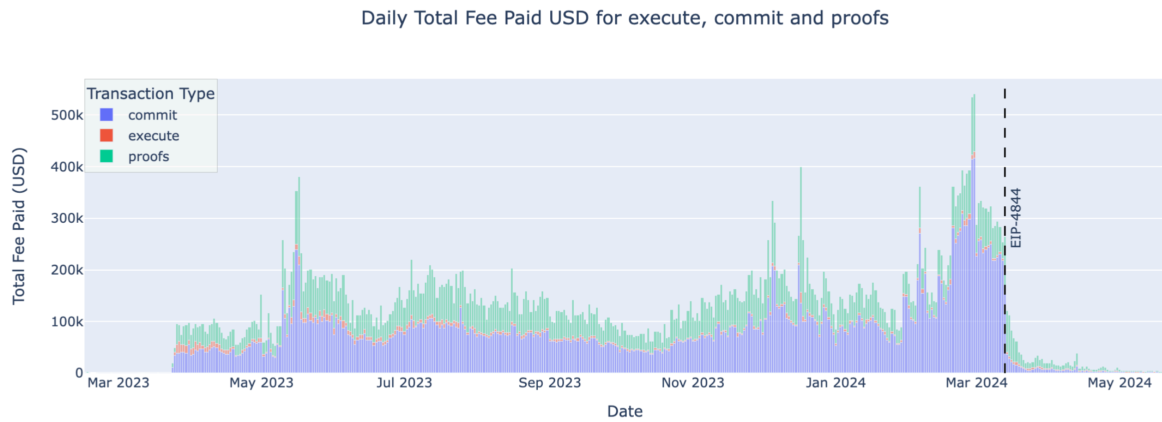


Daily Transactions on Ethereum (for DA and Proofs)

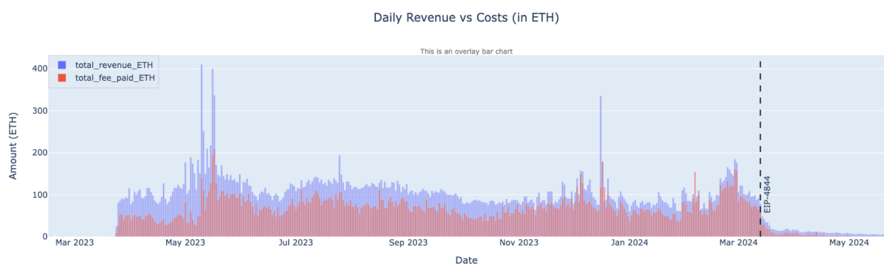


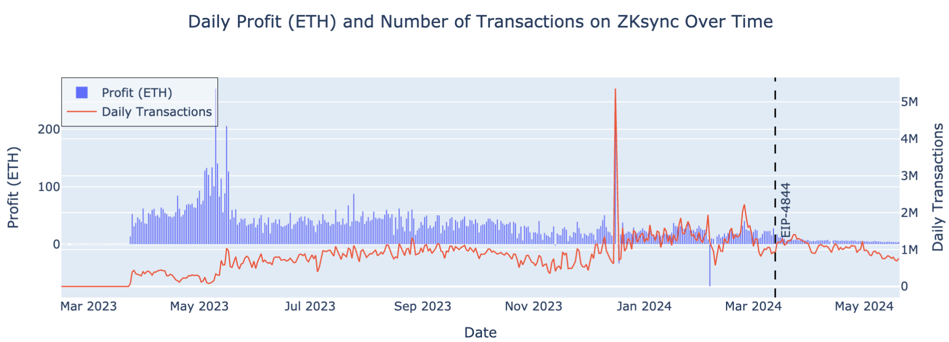
Daily Total Fee Paid (ETH) for execute, commit and proofs



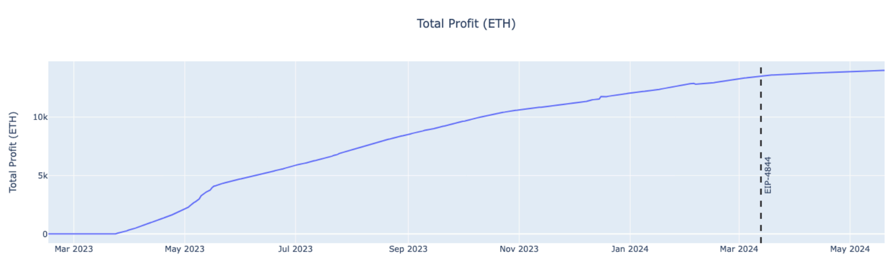


Profitability





We can see lower profits on average than before EIP-4844 at the same number of transactions.



We can see that the profit slows down after EIP-4844.

Conclusion

The analysis of zkSync Era’s transaction fees shows how the system effectively manages costs to improve user experience. Key differences between Ethereum and zkSync, such as intrinsic cost structures and refund mechanisms, optimize resource usage and reduce fees. Batch overhead costs for proving circuits and verifying proofs ensure each transaction fairly contributes to the overall costs, with a refund system for unused resources and overpayments.

After the EIP4844 update, average transaction fees dropped, but profit growth slowed, indicating that demand for transactions remains steady despite lower costs.

Data Tables

	Year_Month	profit_ETH
0	2023-02	-3.98928
1	2023-03	324.96
2	2023-04	1732.14
3	2023-05	2636.86
4	2023-06	1142.86
5	2023-07	1313.2
6	2023-08	1320.2
7	2023-09	1137.21
8	2023-10	986.918
9	2023-11	590.393
10	2023-12	848.908
11	2024-01	743.511
12	2024-02	488.005
13	2024-03	436.824
14	2024-04	194.031
15	2024-05	99.8727

Average Fee Info Before EIP-4844 Implementation (ETH)

Statistic	avg_fee_ETH
mean	0.000282113777646407
std	0.0005494823830349719
min	0
25%	0.00009632506817089792
50%	0.0001305606867561
75%	0.0001926713749024
max	0.0048620504111111

Average Fee Info After EIP-4844 Implementation (ETH)

Statistic	avg_fee_ETH
mean	0.000010391373756484386
std	0.000009284217932952923
min	0.000004200286334465386
25%	0.000007037206838407107
50%	0.000008278282914098034
75%	0.000010835483521301408
max	0.00008655161929978091

Average Fee Info Before EIP-4844 Implementation (USD)

Statistic	avg_sync_fee_usd
mean	0.5160132413387248
std	0.9311224692259991
min	0
25%	0.1797739270723014
50%	0.2571358665076809
75%	0.3848222027560088
max	8.605694677174387

Average Fee Info After EIP-4844 Implementation (USD)

Statistic	avg_sync_fee_usd
mean	0.036185827303090924
std	0.036996900725420945
min	0.01423212187362
25%	0.0221911530365694
50%	0.028101596284786652
75%	0.0367299817854188
max	0.3477638203199302

Descriptive Statistics for DA and Proofs Before EIP-4844

statistic	metric	commit	execute	proofs
total_transactions	mean	1087.359375	32.96134020618557	1184.6752577319587
total_transactions	std	570.9249936017783	56.31418921766638	650.2465663979599
total_transactions	min	1	1	1
total_transactions	25%	675.75	20	767.5
total_transactions	50%	1094.5	31	1262
total_transactions	75%	1553	38	1644.25
total_transactions	max	2216	902	5565
avg_time_diff	mean	2834.6277557957146	4763.109541868753	428.6607288862165
avg_time_diff	std	10298.195709756112	6103.649458010683	1373.966168636385
avg_time_diff	min	38.92960288808664	93.47228381374724	0
avg_time_diff	25%	55.39086807303951	2210.4473684210525	50.972953200645506
avg_time_diff	50%	78.67590148414921	2671.6754032258063	65.35293229916329
avg_time_diff	75%	127.4917658004039	4210.65	102.24821892518693
avg_time_diff	max	55992	67188	11982
avg_fee_paid_ETH	mean	0.04290867337034663	0.09622906848231404	.02376461239858541
avg_fee_paid_ETH	std	0.02765796368459211	0.07531901212619996	.01369938324111557
avg_fee_paid_ETH	min	0.0046842727651678	0.0022487789462354	0.0076069351805565
avg_fee_paid_ETH	25%	0.02526385864269415	0.05617312172751973	.01538829189214082
avg_fee_paid_ETH	50%	0.03288336101787854	0.07816369908197925	0.02061228710832085
avg_fee_paid_ETH	75%	0.0533824466206698	0.11716630454688068	.02882337567308392
avg_fee_paid_ETH	max	0.1506939521364956	0.5997271009635893	0.1081453674373454

Descriptive Statistics for DA and Proofs After EIP-4844

statistic	metric	commit	execute	proofs
total_transactions	mean	247.0183486238532	9.26605504587156	273.78823529411767
total_transactions	std	152.66090367891357	2.2878187971939017	155.78488634114973
total_transactions	min	71	8	75
total_transactions	25%	159	8	185
total_transactions	50%	212	9	244
total_transactions	75%	289	9	301
total_transactions	max	889	19	970
avg_time_diff	mean	444.67165817224065	8645.928968005135	356.19928324818903
avg_time_diff	std	218.38655108752138	1166.773430524165	125.8893588142196
avg_time_diff	min	97.03937007874016	3941.684210526316	88.29278350515465
avg_time_diff	25%	296.2214532871972	8845.333333333334	278.2325581395349
avg_time_diff	50%	404.4339622641509	9034.5	327.1456310679612
avg_time_diff	75%	535.3207547169811	9146.666666666666	447.8918918918919
avg_time_diff	max	1196.7887323943662	9246	654.528
avg_fee_paid_ETH	mean	.00473729604755716	0.03537258606679662	.00814322218876035
avg_fee_paid_ETH	std	0.00457513951005638	.03009100486278729	.00552933842531263
avg_fee_paid_ETH	min	0.0011274148374552	0.0010615794068633	0.0019573492266135
avg_fee_paid_ETH	25%	0.0020292289400612	0.0113003517088008	0.003802146156849
avg_fee_paid_ETH	50%	0.0031641132677449	0.0241517641474285	0.0060427729523001
avg_fee_paid_ETH	75%	0.0051602523819997	0.0500815742505087	0.0110111915390435
avg_fee_paid_ETH	max	0.025088820799811	0.1037143598368376	0.0262903387382806