This is the first part of our three-part series on ERC-4337. In this article, we provide a simple overview of the ERC-4337 standard for account abstraction and compare it to previous approaches to user authentication by analyzing tradeoffs in feature support, censorship resistance, DOS protection, and gas efficiency. In future articles, we will dig into the extensions, and the

costs induced

by the ERC.

TLDR

- Account Abstraction is the ability of a wallet to have arbitrary execution logic. This makes self custodying easy for retail users by enabling multiple options for key management and account recovery.
- ERC-4337 in theory implements censorship-resistant Account Abstraction without making any changes to the consensus layer of Ethereum.
- It achieves this by creating a new mempool, a smart contract wallet standard, and a shared entry point contract for managing User Operations.
- 4337 does however introduce some tradeoffs, namely additional complexity which manifests itself as an increased chance of DOS and a loss in gas efficiency relative to EOA wallets.

Introduction

Self-custodying crypto is HARD. Both new and power users find private key management a nightmare. Moreover, with FTX mishandling customer funds, no retail user will be comfortable storing large sums on an unregulated exchange.

Account Abstraction

(AA) promises to solve the dilemma between key management UX and security for self-custodial funds.

AA achieves this by enabling arbitrary logic to determine if an account can move funds. This enables wallets to have multi-sigs, social recovery,

hardware enclaves

, and much more. Although this has been tried already via

Smart Contract

(SC) wallets, these experiments never gained widespread traction.

Mainly, because users had to rely on a centralizing relay to emit their transactions (Authereum) or hold funds in 2 different wallets (Gnosis Safe), moreover each SC wallet had to develop its own infrastructure increasing development costs.

Newer L2 chains such as zkSync and StarkNet identify the advantages of AA and come prebuilt with it even at the cost of losing compatibility with the EVM spec.

This article intends to explain how ERC-4337 implements AA while making NO changes to the EVM spec.

What does Account Abstraction even mean?

On the Ethereum network, there are currently two types of accounts, External, and SC accounts. AA reduces the two account types down to one, an SC account.

This means that after AA:

In the next section, we will look at the state of the art of a transaction's lifecycle in EVM chains and the constraints coming from the consensus layer.

State of the art

In EVM chains, transactions come from two types of wallets.

EOA wallet’s transaction lifecycle

EOA’s are the most commonly used wallets on EVM chains and consist of the “default” user experience. The transaction lifecycle of an EOA wallet follows several steps each of which has its own checks and constraints to ensure censorship-resistant and economically incentivized inclusion.

In the EVM semantics, EOA accounts interact with dApp smart contracts directly to perform their operations, eg swap on Uniswap or open loans on Compound. The EOA wallets which initiated the transaction are identified through

tx.origin

opcode which returns the first 20 bytes of the public key. While this opcode can be used to identify the EOA, it cannot be used to authenticate a user request as it is carried forward with call context. Smart contract developers typically use

msg.sender

for user authentication purposes and can insure the calling account is an EOA by requiring

msg.sender == tx.origin

.

Transaction lifecycle

Constraints

SC wallet’s transaction lifecycle

As the usage of Ethereum grew, developers grew frustrated at the limitations of EOA wallets and started creating SC wallets with more advanced access control (social recovery, multi-signature, hardware enclave). The transaction lifecycle and the constraints explained above remain the same, but abstraction started happening in the transaction layer.

Transaction lifecycle

Since User Operations cannot be directly submitted to the blockchain, a relayer or an external wallet is required to initiate the transaction. The constraints, imposed by the consensus layer and legacy dApps, are maintained in the above flow, however, it increases the transaction cost.

ERC 4337

Post ERC-4337 any EOA wallet will be able to initiate a transaction containing any User Operation, i.e. it guarantees that the initiator EOA wallet will be paid back for the transaction fees by the User Operation. This is achieved by splitting a User Operation into 2 sandboxed steps. A

validation step

, to check if the User Operation can move funds, and an

execution step

that performs the dApp interaction.

The whole User Operation lifecycle is split into 3 distinct parts:

Bundlers only need to verify if the validation step will succeed, to gain confidence that they will be paid for including this User Operation in their initiated transaction.

User Operation mempool

Censorship resistance and DOS protection

One of the key limitations of current SC wallet implementations of account abstraction is the lack of censorship resistance. While in theory, EOA wallet transactions are censorship-resistant due to the use of the gossip network for message routing, SC wallets generally use centralized message relays, thus being subject to censorship.

The key challenge to achieving censorship resistance is providing DOS protection to the servers forwarding messages. While the Ethereum transaction mempool is far from being considered perfect, the gossip network for EOA accounts is (in theory) protected from DOS by efficiently dropping invalid transactions from the mempool nodes. Each node checks that EOA transactions have:

These checks cost the equivalent of 35k gas to perform on the EVM. You can find a solidity benchmark implemented

[here

](https://github.com/ankitchiplunkar/erc4337#erc4337-gas-estimates----)

.

Since account abstraction enables arbitrary execution logic, work to be performed by mempool to identify invalid User Operations is now a function of the complexity of the validation step and therefore potentially unbounded. Typical SC wallet implementations of account abstractions are therefore forced to use centralized message relayers and achieve DOS protection through traditional means including IP allow / ban lists, API keys, rate limiting, and reputation systems.

ERC-4337 aims to provide the same level of censorship resistance and DOS protection as EOA accounts through the secondary gossip network. To mitigate the DOS vector caused by unbounded compute in the validation step, 4337 imposes a maximum gas limit of 200k gas on

validateUserOp

. Comparing this upper bound to the cost of validating an EOA transaction, we see that it is ~5.5

times more expensive

to perform DOS protection on 4337 transactions than EOA transactions.

This additional overhead on transaction validation along with an unknown number and distribution of nodes in the 4337 gossip network means that there are valid concerns on the ability of 4337 to successfully match the censorship resistance and DOS protection of EOA accounts.

It is worth noting that this compares censorship resistance of 4337 relative to EOA accounts. Both are subject to block-building censorship as seen in MEV-Boost and L2 sequencers.

Generation and inclusion of transaction

Since we are not making changes to the consensus layer, this step remains the same.

Execution of User Operation in a transaction

To ensure that the Bundler address gets paid and User Operation interacts with the dApp as intended, ERC-4337

orchestrates these operations via an EntryPoint contract. The following steps take place in such a transaction:

Gas efficiency

While 4337 moves much of the transaction validation logic on chain, it aims to do so as efficiently as possible in order to provide a good user experience. Since 4337 allows for bundling multiple user operations together in a single EOA transaction, it has the opportunity of amortizing the 21k gas overhead the the EOA across these operations.

The

benchmarks provided by the eth-infinitism implementation

show that in a best case scenario of 10 user operations bundle with simple validation yields an overhead of 39.8k gas per user operation (~2x a typical EOA transaction overhead).

This is not materially different to SC wallets.

Conclusion

In this article, we saw how ERC-4337 attempts to provide censorship-resistant account abstraction and how it compares to legacy transactions from EOA wallets and Smart Contract wallets.

The ERC proposes standards for:

We noted how the additional complexity brought by 4337 introduces some questions on gas efficiency and DOS protection which should be carefully evaluated by teams interested in implementing 4337 in their system.

In the upcoming articles, we will see how the aggregator and paymaster extensions work with ERC-4337 and take a deeper look into the costs associated with making these changes.

Special thanks to

Sid Shekhar

, and Jing Fan for reviewing the article.

References