Account Abstraction on Ethereum

Gabriel-Marius STOICA
National Institute for Research and Development in Informatics - ICI Bucharest
gabriel.stoica@ici.ro

Abstract: There is no doubt that blockchain technology can revolutionize the way we interact and transact goods in both the digital and physical environments. The automatic execution of transactions, without requiring interaction with an intermediary and while recording everything in a publicly available ledger, could be the solution to many current issues, both in the public and private sectors. However, to fully reap the benefits, a series of improvements must be considered and implemented. Even though Ethereum is one of the most popular blockchains at the moment, boasting a large community of developers and users, blockchain technology in general lacks global adoption. There are undoubtedly many aspects that could be improved to bring the next billion users into the web3 space, but most developers agree on one essential topic: the need to improve the user experience by simplifying their interaction with the technical side of the blockchain. Since most users lack technical knowledge, decentralized applications need to abstract this aspect so that it does not constitute a barrier between the benefits and the mode of use. For this reason, Account Abstraction is a proposal that aims to improve the interaction between the end user and decentralized applications.
Keywords: Ethereum, Account Abstraction, Blockchain, Decentralization, ERC-4337.

BACKGROUND

Brief overview of Ethereum

Back in 2013, a programmer named Vitalik Buterin proposed a decentralised, open source blockchain system called Ethereum, enabling the creation and execution of smart contracts. Since then, Ethereum has become one of the most prominent blockchain platforms in the world. Unlike Bitcoin, which is primarily a digital currency, Ethereum is designed to be a decentralized computing platform. Being decentralized means that anyone is allowed to add, but not to remove, data. All participants in the ecosystem, called nodes, agree upon a certain set of new transactions, currently each 15 seconds on Ethereum, which are batched together into a "block".

The key feature in the Ethereum ecosystem, the smart contracts are computer programs living on-chain being capable of handling programmable methods that can be executed later on. The game changer is that once deployed on the blockchain, smart contract code cannot be changed, giving them an immutable character. However, being immutable is not in all cases an advantageous trait and that is why various upgradability patterns have been introduced, such as: diamond pattern (Mudge, 2020), transparent proxy pattern (Palladino, 2018) or universal upgradeable proxy standard (UUPS) (Barros et al., 2019) – which is, currently, the most used one.

https://doi.org/10.54851/v5i2y202304

The most popular examples of smart contracts being deployed so far represent the core of different types of applications such as decentralized trading exchanges (i.e. Uniswap, dYdX, Curve or PancakeSwap), social networks (i.e. Lens Protocol or Farcaster), centralised exchanges (i.e. Binance, Coinbase or Crypto.com), lending apps (i.e. Compound, CoinRabbit or Aave) or Non-Fungible tokens (aka NFTs, i.e. Bored Ape Yacht Club, Crypto Punks or Milady).

As part of the its work method, when any of the above applications are used, or any other transaction is made, the sender needs to pay a fee (usually referred to as gas fee). This fee is an incentive for the participants in the network for processing and verifying transactions. Therefore, the gas fee will be paid in the native token of the blockchain called Ether (ETH).

It's worth saying that on September 15, 2022, Ethereum went through "The Merge" upgrade which transitioned Ethereum from proof-of-work to proof-of-stake.

The Merge represented Ethereum's most significant enhancement, slashing its energy usage for network security by an astounding 99.95%. This transformation not only fortified Ethereum's network but also substantially diminished its carbon footprint. Consequently, Ethereum has elevated its security and scalability, positioning itself as an eco-friendly blockchain (Ethereum, 2023).

In order to interact with the Ethereum ecosystem, an Ethereum account is needed. As a user, you are going to be represented on-chain by a unique wallet address, being able to read your balance, send transactions and connect to applications. That is why we need to understand what's the current account model, how it works in the Ethereum ecosystem and how it can be improved.

The current account model on Ethereum

When we are speaking about the Ethereum's account model, we can see it as a state machine. Every transaction modifies the state of this machine, which holds information about millions of accounts (currently 240.41 million) (Ycharts, 2023).

It is important to understand that on the Ethereum ecosystem there are two types of accounts:

- Externally Owned Accounts (EOAs): these accounts are controlled by private keys and have no associated code, being typically used by normal users or entities to send transactions, which can include ETH transfers or triggering smart contract functions. Any Externally Owned Account is identified by a cryptographic pair of public and private keys, thus, when speaking about EOAs, everything comes to how we store and manage private keys. Therefore, there are multiple types of wallets being capable of handling even multiple EOAs for you, such as:
  - Hardware wallets: one of the most secure solutions when comes to storing and accessing your assets, i.e., Ledger, Trezor or SafePal;
  - Software wallets: the most used type of wallets, while they offer convenience, they are generally considered less secure than hardware wallets, especially if the device they are on is compromised, i.e., MetaMask, Trust Wallet or Coinbase Wallet;
  - Key Management Solutions (KMS): for enterprises or users with significant assets, they provide robust security features, i.e. AWS Key Management Service, Azure Key Vault or HashiCorp Vault.
- Contract Accounts are deployed as a smart contract; unlike EOAs, are governed by their associated code. The code, written in high-level languages like Solidity or Vyper, gets compiled down to Ethereum Virtual Machine (EVM) bytecode. As a result, we can highlight some key differences:
  - Stateful: each contract account has storage space where the state information is maintained. This feature allows to remember data between function calls and transactions;
  - No direct control: contract accounts do not have private keys, as a result their behaviour is entirely determined

by their code, being able to "act" or "respond" depending on the message or transaction received;
- Governed by code: as stated before, a contract account operates based on the logic defined in the associated code. They cannot initiate actions on their own but can execute functions when requested.

Each Ethereum account has four fields used when interacting with the Ethereum ecosystem (Cook, 2023):
- nonce: a counter that indicates the number of transactions sent from an EOA or the number of contracts created by a contract account;
- balance: the number of wei owned by this address. Wei is a denomination of ETH and there are 10^{18} wei per ETH;
- codeHash: a hash value computed over the code fragments of a contract account, unable to be changed. Since EOAs cannot have associated code, this field is the hash of an empty *string*;
- storageRoot: the storage hash is a 256-bit identifier derived from a specific type of tree structure called the Merkle Patricia tree. This tree represents the account's stored data, mapping unique keys to their respective values. By default, this tree is empty.

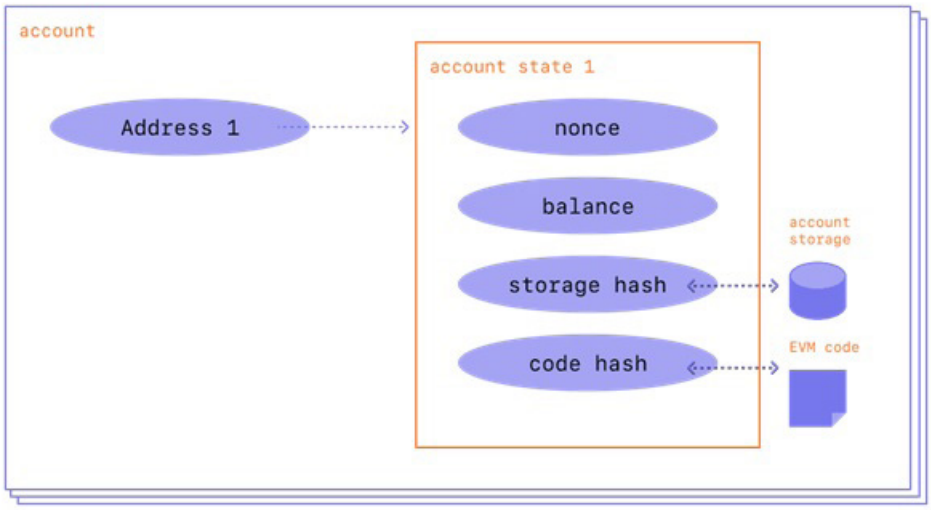

Figure 1. *Diagram of Ethereum Account State Model (Takenobu 2018)*

While revolutionary in its design and capabilities, the Ethereum ecosystem is far from being widely adopted. There are many limitations consisting of impediments for non-technical users, preventing broad adoption and onboarding the next billion users, such as:
- Key management is hard: being coupled with a single key, the most common negative occurrence among crypto users is the loss or theft of the private key. As a result, the cryptographic pair is hard to secure and recover;
- Multiple limitations of EOAs:
  - one size fits all mechanism: you do not have any granular control;
  - gas payments: each EOA must mai\ ntain its ETH balance to pay for the gas fees, making it hard to manage for non-technical users;
  - no privacy: since the most frequent source of funds are Centralised Exchanges (CEXs), your identity is attached directly to your EOA; it is not hard to track data connected to a user

account (i.e. CEXs have KYC – Know Your Customer processes); on this concern, there are multiple solutions on the market at the moment trying to solve this issue, such as Omnia (omniatech.io);
- No way to batch multiple transactions: when required to create and send multiple transactions, each of them must be signed and executed separately, meaning that you will have to pay the gas fee twice.

Having everything in mind, it is obvious that the Ethereum ecosystem needs a solution to fix all of these issues and enhance the user experience.

ACCOUNT ABSTRACTION

Introduction to Account Abstraction

One of the proposed solutions to address the limitations listed above and enhance flexibility is the concept of Account Abstraction. The primary motivation behind Account Abstraction is to simplify the Ethereum account model by treating all accounts uniformly. Instead of having separate rules and behaviors for EOAs and Contract Accounts, Account Abstraction proposes a unified model where the distinction is blurred, and all accounts can be represented as contracts.

Basically, each Ethereum account could be defined and managed by a special-purpose smart contract, allowing wallet developers to craft systems capable of handling processes like purchasing funds, taking care of key management and recovery, gas payment or arbitrary access control mechanism.

Advantages of Account Abstraction

As stated before, the Account Abstraction mechanism comes with a lot of advantages making it much easier for the end-user to interact with and benefit from blockchain technology.

At the heart of Account Abstraction is the idea of treating all accounts, whether Externally Owned Accounts or Contract Accounts, as contracts. This means that the traditional distinction between these two account types is blurred, and every account can be represented and managed by its own contract logic.

Traditionally, acquiring funds in the Ethereum ecosystem required users to navigate external exchanges or platforms (i.e. Centralized Exchanges such as Binance or Coinbase or Decentralized Exchanges such as Uniswap or dYdX). With Account Abstraction, wallet developers can integrate direct purchasing mechanisms within the wallet itself. For example, a user could seamlessly buy ETH or other tokens directly from their wallet interface, bypassing the need for third-party services and potentially benefiting from reduced fees or faster transaction times.

Additionally, the loss or compromise of private keys has been a significant concern in the blockchain domain from the beginning. Account Abstraction allows for more advanced and user-friendly key management solutions. A wallet could, for instance, implement a multi-factor authentication system or a biometric verification process to recover access to funds, reducing the dependence on easily lost recovery phrases. Another possibility is to introduce a social recovery mechanism, allowing a user to assign several trusted individuals as recovery representatives.

On top of that, another improvement would be to sponsor the gas fees paid by the user to interact with a dApp. In other others, developers could choose, as part of their tokenomics model, to subsidize the gas fees paid for each transaction on their platform, allowing the end-user to simply use the service as it is without worrying about maintaining all the time the ETH balance. Or imagine a scenario where a user, holding multiple tokens but no ETH, wishes to execute a transaction through the same dApp. Under this new model, they could potentially pay the gas fee using an alternative token, like DAI or USDC, enhancing user experience and economic fluidity on the platform.

The traditional method of access control in Ethereum has been relatively straightforward, relying primarily on private key ownership. Account Abstraction opens the door for more

intricate and tailored access controls. For instance, a business wallet could be designed to require multiple signatures from different department heads for high-value transactions, ensuring checks and balances, this mechanism being known as multi-signature. Alternatively, a user could set up a time-lock mechanism, where funds can only be withdrawn after a certain date, serving as a self-imposed savings mechanism.

HISTORY OF ACCOUNT ABSTRACTION PROPOSALS

Account abstraction has been a topic of interest within the Ethereum community for several years. Over the years, several Ethereum Improvement Proposals (EIPs) have been introduced to address the challenges and implement an account abstraction mechanism.

EIP-86: Abstraction of transaction origin and signature

EIP-86 also known as „Abstraction of transaction origin and signature", was one of the earliest proposals related to account abstraction. It aimed to make the Ethereum transaction format more flexible by allowing users to create transactions without a *signature*. This would enable the use of smart contracts to handle transaction validation, thereby abstracting away the need for Externally Owned Accounts to initiate transactions (Ethereum, 2021).

The proposal suggested removing the signature component from transactions and replacing it with an *init* field. This would allow users to create new contract accounts by providing an initialization code. Additionally, the proposal introduced a new way to pay for gas fees using any token, not just ETH.

EIP-2938: Account Abstraction via changes to Ethereum protocol

EIP-2938 proposed a more comprehensive approach to account abstraction. It aimed to make smart contracts first-class citizens in the Ethereum ecosystem, allowing them to initiate transactions and interact with other contracts directly (Buterin et al., 2020).

The proposal introduced a new transaction type, *AA_TX_TYPE*, that included fields like nonce, *target*, and *data*. The *target* would be the entry point contract address, and *data* would contain the EVM bytecode.

Thus, EIP-2938 highlighted the potential benefits of account abstraction, such as improved security, better user experience, and the ability to pay gas fees in any token. However, it also underscored the challenges and complexities involved in implementing such a system, requiring changes to the Ethereum consensus layer. In other words, to integrate EIP-2938, a hard fork of the Ethereum network was required.

EIP-3074: AUTH and AUTHCALL opcodes

EIP-3074 proposed the introduction of two new EVM opcodes, *AUTH* and *AUTHCALL*, to improve the flexibility of account interactions. The *AUTH* opcode would allow for the extraction of an Ethereum address from a signed message (using the private key of the owner), while the *AUTHCALL* opcode would enable calls to be made with the authorized address as the *msg.sender* (the EOA address). This would allow for more flexible transaction initiation and delegation patterns (Wilson et al., 2020).

Therefore, EIP-3074 aimed to simplify user experiences by enabling operations like batched transactions, sponsored transactions, and more. It provided a way to delegate transaction execution without the need for complex smart contract logic.

ERC-4337: Account Abstraction using alt mempool

ERC-4337 is a recent proposal that seeks to implement account abstraction without requiring consensus-layer protocol changes, comparing to all the other implementations discussed before. Instead, it introduces a higher-layer pseudo-transaction object called a *UserOperation*. Users send *UserOperation* objects to a separate mempool, which can be

seen as a traditional transaction, but with a few key differences (Buterin et al., 2021):

- Dedicated mempool: as stated above, UserOperations are sent to a different mempool than the one used for all other types of transactions, waiting to be packed by the Bundlers;
- Extra fields: UserOperations include new fields in the transaction structure, e.g., EntryPoint, Bundler, Paymaster and Aggregator (Alchemy, 2023);
- Intent based: currently, user's intent is specific, i.e., swap a specific amount of USDC for a specific amount of ETH. Instead, ERC-4337 allows users to trade an amount of USDC for the most amount of ETH possible (Alchemy, 2023).

Bundlers, a special class of actors, represents a type of Ethereum node capable of processing UserOperations which are dispatched to a collective of Bundlers. These Bundlers oversee the mempool, consolidating several UserOperations into a singular transaction. They then process and relay these bundled transactions to the blockchain for the users. As a reward for their services, Bundlers receive compensation from the users.

When relaying, Bundlers send all transaction to a singleton smart contract called EntryPoint, which verifies and executes UserOperations as follows:

- Verify whether the wallet possesses sufficient funds to cover the potential maximum gas usage (as indicated by the gas field in the UserOp). If it falls short, the transaction is declined;
- Execute the transaction, taking money from the Smart Contract Account (SCA) to reimburse the Bundler.

The key feature proposed by the ERC-4337, is a concept of Smart Contract Account (SCA), which, at its core, is a fully programmable smart contract, being able to interact with other smart contracts deployed on-chain. Instead of being controlled by a private key, as a EOA, SCAs are controlled by arbitrary logic which can be customized before its deployment.

For instance, SCAs can implement custom logic for social recovery, in case the user loses access, or multi-sign approach, allowing multiple users controlling it (Alchemy, 2023).

The ERC-4337 can also support batched transactions, through a dedicated operator called Aggregator. The Aggregator can combine multiple signatures into a single one, enabling validation of multiple bundled UserOperations in a single step.

To support the automatically gas payment feature, ERC-4337 comes with a component called Paymaster. Therefore, paymasters allow developers to:

- Enable gas payments in stablecoins;
- Enable gas payments in other ERC-20 tokens;
- Sponsor gas fees for their users (Alchemy, 2023).

The approach exposed above avoids the need for consensus-layer changes and allows for more flexible account interactions. Because it runs on top of the blockchain, hence not requiring any changes to the protocol, this makes it usable as it is and that is why it was rolled out to Ethereum mainnet on March 1st, 2023, at the WalletCoin conference in Colorado.

It is worth mentioning that the ERC-4337 standard can be deployed on any EVM compatible blockchain, such as Polygon, Binance Smart Chain, Arbitrum, Optimism or Avalance, expanding its usability across multiple networks.

IMPLICATIONS AND FUTURE DIRECTIONS

Potential impact of AA on Ethereum

As previously described, the introduction of Account Abstraction into the Ethereum ecosystem promises to usher in a new era of flexibility and innovation. From a user's standpoint, AA can significantly enhance the overall experience on the Ethereum platform, no longer bound by the traditional constraints of Externally Owned Accounts. Users can benefit from more personalized security measures, streamlined transaction processes, and the

potential to pay gas fees with tokens other than ETH or even have custom gas payments to pay for. This not only simplifies interactions with the blockchain but also offers a more intuitive and user-centric approach, bridging the gap between the complexities of blockchain technology and the everyday user.

For developers, AA is akin to being handed a new set of tools with which they can craft more intricate and tailored applications. By treating all accounts as contract-like entities, developers can embed custom logic directly into accounts, allowing for innovative transaction validation mechanisms and advanced multi-signature schemes, as described in the article. This level of customization can lead to the creation of applications that are more secure, efficient, and aligned with specific user needs.

Future developments and improvements

As Ethereum has already rolled out Account Abstraction (AA), it is exciting to think about what comes next. The first steps with AA will teach us a lot, but that is just the start of a bigger journey. First up, we need to make sure AA is safe. With this new way of managing accounts, we must be extra careful to avoid any security issues. Regular checks and feedback from the Ethereum community will be key to spotting and fixing any problems.

For everyday users, the aim is to make AA easy to use. This means designing simple apps, offering clear guides, and providing tools that make the whole process straightforward. We want everyone, even those new to blockchain, to enjoy the perks of AA without any hassle.

For developers, the future looks bright. With AA in their toolkit, they can come up with even cooler features. To help them out, Ethereum plans to offer more resources, like advanced tools and helpful guides, so they can make the most of AA.

Currently, there are a lot of web3 companies building on top of Account Abstraction, such as:

- Smart contract wallets: Argent, Safe, Ambire, blocto;
- Applications: dYdX, Lens, Sorare;
- Bundlers/Relayers: OpenZeppelin, Biconomy, Stackup.

With that being said, another big goal is to make sure AA works well with other tech solutions and even other blockchains. As the world of blockchain grows, it is important for AA to easily connect and work with different systems.

ACKNOWLEDGMENT

The research for this article is conducted within the project „Resilient and Interoperable Communication Systems based on Distributed Technologies and Self-Sovereign Digital Identity (RoDID),” funded by the Advanced Research Program based on Emerging and Disruptive Technologies - Support for the Society of the Future (FUTURE TECH).

REFERENCE LIST

Alchemy. (2023) *How ERC-4337 supports Account Abstraction*. https://www.alchemy.com/learn/account-abstraction

Barros, G., Gallagher, P. (2019) *ERC-1822: Universal Upgradeable Proxy Standard (UUPS).* https://eips.ethereum.org/EIPS/eip-1822

Buterin, V., Dietrichs, A., Garnett, M., Villanueva M., Wilson S. (2020) *EIP-2938: Account Abstraction.* https://eips.ethereum.org/EIPS/eip-2938

Buterin, V., Weiss, Y., Tirosh, D., Nacson, S., Forshtat, A., Gazso, K., Hess, T. (2021) *ERC-4337: Account Abstraction Using Alt Mempool.* https://eips.ethereum.org/EIPS/eip-4337

Cook, J., Freeborn, C., Smith, C., Joshua (2023) *Ethereum Accounts.* https://ethereum.org/en/developers/docs/accounts/

Ethereum Foundation. (2021) *EIP-86*. https://github.com/ethereum/EIPs/blob/master/EIPS/eip-86.md

Ethereum Foundation. (2023) *What is Ethereum*. https://ethereum.org/en/what-is-ethereum/

Mudge, N. (2020) *ERC-2535: Diamonds, Multi-Facet Proxy.* https://eips.ethereum.org/EIPS/eip-2535

Palladino, S. (2018) *The transparent proxy pattern.* https://blog.openzeppelin.com/the-transparent-proxy-pattern

Takenobu T. (2018) *Ethereum EVM illustrated*. https://takenobu-hs.github.io/downloads/ethereum_evm_illustrated.pdf

Wilson, S., Dietrichs, A., Garnett, M., Zoltu, M. (2020) *EIP-3074: AUTH and AUTHCALL opcodes.* https://eips.ethereum.org/EIPS/eip-3074

YCharts (2023) *Ethereum Cumulative Unique Addresses (I:ECUA).* https://ycharts.com/indicators/ethereum_cumulative_unique_addresses