The Smart Account Revolution: A Comprehensive Analysis of Programmable Blockchain Accounts, ERC-4337, and the Future of Web3

Section 1: The Evolution of Blockchain Accounts: From EOAs to Smart Accounts

The history of blockchain interaction is fundamentally a story about accounts. In the Ethereum ecosystem and its many derivatives, the account is the user's anchor—a digital identity that holds assets, executes transactions, and interacts with a burgeoning world of decentralized applications (dApps). For years, this interaction was defined by a single, rigid model: the Externally Owned Account (EOA). While simple and powerful in its own right, the EOA model carries inherent limitations that have created a significant barrier to mainstream adoption. In response, a paradigm shift is underway, moving from these static, key-based accounts to dynamic, code-based "Smart Accounts." This evolution, underpinned by the principle of Account Abstraction, represents one of the most critical infrastructure upgrades in the pursuit of a more secure, usable, and accessible Web3.

1.1. Understanding Traditional Blockchain Accounts: Externally Owned Accounts (EOAs)

On the Ethereum network, there are two primary types of accounts: Externally Owned Accounts (EOAs) and Contract Accounts.[1] The EOA is the most common type, often referred to simply as a "wallet" by users. It is the standard account model used by popular wallets like MetaMask, Ledger, and Trust Wallet.[2]

Core Mechanics

An EOA is defined and controlled by a cryptographic keypair: a public key and a private key.[1] The public key is used to generate the public address (e.g., a

0x... address) that other users interact with to send funds or assets. The private key, often represented to the user as a 12 or 24-word "seed phrase" or "secret recovery phrase," grants absolute and indivisible control over the account.[3] Anyone who possesses the private key can sign any transaction on behalf of the account, giving them full authority to transfer assets or interact with smart contracts.[4]

These accounts are termed "Externally Owned" because their control originates from outside the blockchain's state machine; they are controlled by a user holding a private key, in contrast to Contract Accounts, which are controlled by their own internal code.[6] An EOA itself does not contain any executable code. Its on-chain representation consists of only two pieces of data: its Ether (ETH) balance and a nonce, which is a transaction counter used to prevent replay attacks.[1] Every transaction on the Ethereum network must originate from an EOA, as only EOAs have the protocol-level ability to initiate an action by signing it with a private key.[5]

Inherent Limitations & The User Experience "Glass Ceiling"

The simplicity of the EOA model is, as some have described it, a "double-edged sword".[8] While it provides a direct and straightforward way to interact with the blockchain, its rigidity creates a series of critical limitations that collectively form a "glass ceiling" for user experience and security, hindering the transition from a niche technology to a mainstream platform.

- **Single Point of Failure:** The most significant flaw of the EOA model is its reliance on a single private key for security. The seed phrase becomes the ultimate secret that must be protected at all costs. If this key is lost, stolen, or otherwise compromised, all funds and assets associated with the account are permanently and irreversibly lost.[3] There is no "Forgot Password" button, no customer support line to call, and no recovery mechanism built into the protocol.[8] This places an immense and often unforgiving burden of key management on the user, a stark contrast to the safety nets provided by traditional financial systems.[10]
- **Rigid Transaction Logic:** EOAs are limited to performing one atomic operation per transaction. Complex interactions with dApps, which are common in areas like Decentralized Finance (DeFi), often require multiple steps. For example, swapping one

token for another on a decentralized exchange typically requires two separate transactions: first, an approve transaction to grant the exchange's smart contract permission to spend the token, and second, a swap transaction to execute the trade.[4] Each of these steps requires a separate signature from the user, creating a clunky, time-consuming, and often expensive user flow.[9]

- **No Programmable Security:** The hard-coded logic of EOAs prevents the implementation of sophisticated, on-chain security features that users of modern financial applications take for granted. It is impossible to natively enforce rules such as multi-signature requirements (requiring multiple parties to approve a transaction), daily spending limits, whitelisting of trusted recipient addresses, or time-locked withdrawals.[3] This results in a rudimentary, "one-account-fits-all" security model that is ill-suited for the diverse needs of individuals, teams, and organizations.[15]
- **Gas Fee Inflexibility:** Transactions initiated from an EOA must have their associated "gas" fees (the cost of computation on the network) paid in the blockchain's native token—ETH on Ethereum, MATIC on Polygon, and so on.[16] This forces users to constantly acquire and maintain a balance of a specific, often volatile, asset simply to interact with the network, even if their primary goal is to use stablecoins or other tokens. This is a major point of friction, especially for new users who may not own the native token and must go through the extra step of acquiring it from an exchange.[5]

The core conflict in blockchain account design has long been between the absolute self-custody offered by EOAs and the user-friendly experience demanded by mainstream audiences. EOAs maximize user sovereignty, embodying the "not your keys, not your crypto" ethos, but they do so at a steep cost to usability and practical security for anyone who is not a technical expert. This fundamental tension has driven the search for a new model that can provide a more forgiving and intuitive experience without sacrificing the core principles of decentralization.

1.2. The Paradigm Shift: Introducing Smart Accounts

In response to the limitations of EOAs, the blockchain community has developed a new type of account: the Smart Contract Account (SCA), more commonly known as a "Smart Account".[2] These accounts represent the next evolution in blockchain wallet technology, moving beyond the static private key model to one that is dynamic, flexible, and programmable.[4]

Defining Smart Accounts (SCAs)

A Smart Account is a blockchain account that is itself a smart contract, managed by its own embedded code rather than by an external private key.[1] This fundamental architectural difference is the source of all its advantages. By making the account's logic programmable, developers can embed custom rules for transaction authorization, security policies, and key management directly into the account.[3] Ethereum co-founder Vitalik Buterin has identified the transition to smart accounts as a major and necessary evolution for wallet security, arguing that without it, users facing the complexities of EOA management will increasingly opt for the custodial convenience of centralized exchanges, undermining the very principle of decentralization.[14]

The Core Principle: A Conceptual Overview of Account Abstraction

The enabling technology behind smart accounts is **Account Abstraction (AA)**. In the context of Ethereum, account abstraction is the concept of "abstracting" away the protocol's hard-coded rules for account behavior—specifically, the requirements for transaction validation (signature verification) and execution—and allowing these rules to be defined by flexible smart contract code.[4]

Instead of the protocol dictating that a valid transaction must be signed by a single ECDSA private key, account abstraction allows an account to define its own validity conditions. This could mean requiring signatures from multiple keys (multisig), a signature from a more modern cryptographic algorithm (like those used in smartphones), or even no signature at all under certain conditions (e.g., for very small, pre-approved transactions).[4] In essence, account abstraction allows developers to programmatically define:

1. **Authentication:** *Who* is allowed to authorize an action on behalf of the account.
2. **Authorization:** *What* actions they are permitted to perform and under what conditions.[4]

This unification of Ethereum's two account types into a single, programmable model is widely seen as a prerequisite for onboarding the next billion users to Web3, as it provides the foundation for building experiences that are as secure and intuitive as the best Web2 applications.[9] Smart accounts are the practical implementation of this powerful concept.

Table 1: EOA vs. Smart Contract Account (SCA) - A Comparative Overview

Feature	Externally Owned Account (EOA)	Smart Contract Account (SCA) / Smart Account
Control Mechanism	Controlled by a single, external private key (seed phrase).[3]	Governed by the programmable code of the smart contract itself.[1]
Creation Cost	Free to generate a keypair off-chain; no on-chain cost until the first transaction.[5]	Requires an on-chain transaction to deploy the contract, incurring a gas fee.[3]
Transaction Initiation	Can directly initiate transactions on the blockchain.[5]	Cannot initiate transactions directly; requires an EOA relayer (e.g., a Bundler in ERC-4337) to start the process.[1]
Key Management	Rigid; the private key is immutably linked to the address, creating a single point of failure.[4]	Flexible; supports key rotation, multi-signature setups, and social recovery mechanisms.[14]
Security Features	Rudimentary; no native support for spending limits, whitelists, or account freezing.[3]	Programmable; allows for custom security policies like multisig, spending limits, time-locks, and whitelists.[11]
Transaction Features	Limited to a single operation (e.g., one token transfer or contract call) per transaction.[4]	Supports atomic transaction batching, combining multiple operations into a single on-chain transaction.[2]
Gas Payments	Inflexible; fees must be paid in the network's native token (e.g., ETH).[5]	Flexible (Gas Abstraction); fees can be sponsored by a third party or paid in various ERC-20 tokens.[14]

Account Recovery	Impossible; if the private key is lost, the account and its assets are permanently inaccessible.[3]	Possible; can be designed with social recovery mechanisms using trusted guardians.[2]

Section 2: ERC-4337: The Technical Blueprint for Account Abstraction

While the concept of account abstraction has been discussed for years, its practical implementation on Ethereum was stalled by the significant challenge of modifying the core protocol. A direct, "native" implementation would require a hard fork—a complex and contentious network-wide upgrade. To circumvent this, a team of Ethereum developers, including Vitalik Buterin, proposed **ERC-4337**, a standard that achieves account abstraction without any changes to the consensus layer.[14] Deployed in March 2023, ERC-4337 provides a standardized blueprint for smart accounts, creating a new, higher-level infrastructure that runs parallel to Ethereum's existing transaction system.[22]

2.1. A Pragmatic Compromise: Why ERC-4337?

The core genius of ERC-4337 lies in its design as a pragmatic compromise. It recognized that altering the fundamental transaction type of Ethereum was a monumental task that would break countless existing applications and wallets.[23] Instead of changing the rules at the base layer, ERC-4337 builds a new system on top of it.[24] It effectively replicates the functionality of Ethereum's transaction mempool in a higher-level, smart contract-based system.[21] This "overlay protocol" approach means that ERC-4337 is not a core part of Ethereum itself but a standard that anyone can use on any EVM-compatible blockchain today, from Ethereum mainnet to Layer 2 networks like Polygon, Arbitrum, and Optimism.[17] It cleverly simulates native account abstraction, providing a user-facing illusion of a new transaction paradigm while still relying on the underlying EOA-based system to ultimately get transactions included in a block.

2.2. A New Transaction Lifecycle: The UserOperation

At the heart of the ERC-4337 standard is a new data structure called the UserOperation.[26] This is not a standard Ethereum transaction but a "pseudo-transaction object" that encapsulates a user's

intent to perform an action from their smart account.[21] It contains all the necessary information for the network to process the user's request, including fields analogous to a regular transaction (like

nonce and callData) but also new fields that enable the advanced features of account abstraction.[27]

Key fields within a UserOperation include:

- sender: The address of the smart account initiating the operation.
- nonce: An anti-replay value specific to the smart account.
- initCode: The bytecode used to create the smart account if it hasn't been deployed yet (for counterfactual deployment).
- callData: The data for the function call to be executed by the smart account (e.g., the details of a token transfer or swap).
- gas fields (callGasLimit, verificationGasLimit, preVerificationGas, maxFeePerGas, maxPriorityFeePerGas): A more granular set of gas parameters to manage the costs of the two-phase validation and execution process.
- paymasterAndData: An optional field specifying the address of a Paymaster contract that will sponsor the gas fees, along with any required data for that Paymaster.
- signature: The signature that proves the UserOperation was authorized by the smart account's owner(s), according to its custom validation logic.

2.3. The ERC-4337 Infrastructure: Key Components and Their Roles

ERC-4337 defines a new, decentralized infrastructure composed of several key on-chain and off-chain components that work in concert to process UserOperations.

The Alternate Mempool

Instead of being broadcast to the main Ethereum transaction mempool, signed UserOperation objects are sent to a separate, off-chain P2P network, often called the "alt mempool".[20] This is where they await processing by the next component in the chain: Bundlers.

Bundlers: The Off-Chain Orchestrators

Bundlers are the workhorses of the ERC-4337 system. They are specialized nodes, run by any entity, that must possess a traditional EOA with a balance of ETH.[24] Their primary functions are:

1. **Monitoring:** They listen to the alt mempool for incoming UserOperations.[20]
2. **Validation & Simulation:** They simulate each UserOperation off-chain by calling a special function on the EntryPoint contract to verify its validity and ensure that processing it will be profitable.[31]
3. **Bundling:** They select multiple valid UserOperations and package them into a single, standard Ethereum transaction called a "bundle".[24]
4. **Submission:** They submit this bundle transaction to the main Ethereum network by calling the EntryPoint contract, paying the required gas fee in ETH from their own EOA.[20]

Bundlers are economically incentivized to perform this service. They are reimbursed for the gas they spend, plus a priority fee (a tip), which is paid by the user's smart account or a Paymaster. This creates a permissionless, competitive market for bundling services, aligning incentives to ensure UserOperations are processed efficiently.[24]

The EntryPoint Contract: A Universal On-Chain Gateway

The EntryPoint is a global, trusted singleton smart contract that serves as the central orchestrator and single point of entry for all ERC-4337 transactions on a given chain.[20] It is a highly audited and secure contract that Bundlers interact with. Its core responsibility is to safely validate and execute the bundles of

UserOperations it receives.

The EntryPoint operates in a distinct two-phase process to ensure that Bundlers are always compensated for their work, even if a user's intended action fails:

1. **Verification Loop:** The EntryPoint iterates through each UserOperation in the bundle and calls the validateUserOp function on the corresponding smart account. This function contains the account's custom logic to verify the signature and other parameters. If a Paymaster is involved, the EntryPoint also calls validatePaymasterUserOp on the

Paymaster contract. During this phase, it collects the necessary gas fees from the accounts or Paymasters and holds them in escrow.[24]

2. **Execution Loop:** After successfully verifying all operations, the EntryPoint proceeds to the execution loop. It iterates through each UserOperation again and executes its callData on behalf of the smart account—for example, calling the Uniswap router to perform a token swap.[24]

This separation is critical: if an operation's execution fails (e.g., the swap reverts due to high slippage), the verification has already succeeded, and the EntryPoint can still use the collected fees to pay the Bundler. This protects Bundlers from griefing attacks where users submit operations that are designed to fail after consuming gas.[35]

Paymasters: Abstracting Gas Payments

Paymasters are optional smart contracts that introduce gas payment flexibility.[20] When a

UserOperation specifies a Paymaster, that contract can agree to pay the gas fees on the user's behalf. This enables two powerful use cases:

- **Sponsored Transactions:** A dApp can deploy a Paymaster to cover the gas costs for its users, creating a "gasless" experience that dramatically improves onboarding.[15]
- **ERC-20 Gas Payments:** A Paymaster can be designed to accept payment from the user in an ERC-20 token (like USDC), swap it for ETH behind the scenes, and then pay the EntryPoint in ETH. This frees the user from needing to hold the network's native token.[38]

Paymasters must stake ETH with the EntryPoint contract to participate in the system. This stake acts as a bond to disincentivize malicious behavior, as a Paymaster that causes bundles to fail can be throttled or banned by Bundlers.[38]

Wallet Factories: Deterministic Account Creation

New smart accounts are not created directly but are deployed by designated "Factory" contracts.[26] ERC-4337 leverages the

CREATE2 opcode, which allows for "counterfactual deployment." This means the address of a new smart account can be calculated deterministically off-chain before the account contract is actually deployed on-chain.[7] A user can be given this address, receive assets to it, and the factory contract will only be called to deploy the account's code during the processing of its very first

UserOperation.[35] This saves gas and streamlines the onboarding process.

2.4. The Complete Workflow: From User Intent to On-Chain Execution

The entire ERC-4337 lifecycle can be summarized in the following steps [24]:

1. **Creation:** A user, interacting with a dApp, crafts a UserOperation object that specifies their desired action (e.g., "swap 100 USDC for ETH on Uniswap").
2. **Signing:** The user signs the hash of the UserOperation using the authentication method defined by their smart account (e.g., a private key, a passkey, etc.).
3. **Submission to Mempool:** The signed UserOperation is sent to the off-chain alt mempool, where it is visible to Bundlers.
4. **Bundling:** A Bundler selects the UserOperation from the mempool, simulates its validation to ensure it's valid and profitable, and includes it in a bundle with other UserOperations.
5. **On-Chain Call:** The Bundler creates a standard Ethereum transaction that calls the handleOps function on the global EntryPoint contract, passing the array of bundled UserOperations as an argument. The Bundler pays the gas for this transaction in ETH.
6. **Verification:** The EntryPoint contract receives the bundle and begins the verification loop. For each UserOperation, it calls the validateUserOp function on the specified sender (the smart account). If a Paymaster is used, its validatePaymasterUserOp function is also called. The EntryPoint collects the required gas fees.
7. **Execution:** After successful verification of all operations, the EntryPoint begins the execution loop, calling the target contract with the callData for each UserOperation.
8. **Reimbursement:** The EntryPoint calculates the actual gas consumed by each operation and reimburses the Bundler from the funds collected during the verification step.

Table 2: Key Components of the ERC-4337 Standard

Component	Type	Primary Role	Key Interaction
UserOperation	Data Structure	Represents a user's	Sent by the user to

		transaction intent.[21]	the alternate mempool.
Bundler	Off-Chain Actor	Packages UserOperations into a bundle and submits it to the blockchain.[20]	Monitors the mempool and calls handleOps on the EntryPoint.
EntryPoint	On-Chain Contract	A trusted singleton that verifies and executes bundles of UserOperations.[24]	Called by Bundlers; calls validateUserOp on Smart Accounts.
Paymaster	On-Chain Contract	An optional contract that sponsors gas fees or allows payment in ERC-20 tokens.[27]	Called by the EntryPoint during the verification phase.
Wallet Factory	On-Chain Contract	Deploys new smart account contracts deterministically.[7]	Called by the EntryPoint to deploy an account during its first UserOperation.

Section 3: Unlocking Value: Core Features and Use Cases of Smart Accounts

The complex machinery of ERC-4337 is not an end in itself; it is the means to unlock a suite of powerful features that fundamentally reshape the user experience and security landscape of Web3. These features are not merely incremental improvements but are direct solutions to the most significant pain points of the traditional EOA model. They achieve this by effectively porting familiar, intuitive concepts from the Web2 world into a decentralized, trustless context.

3.1. Revolutionizing User Experience (UX)

The most immediate and impactful benefits of smart accounts are centered on abstracting away the complexities that have historically plagued blockchain interactions.

Gas Abstraction: Gasless & ERC-20 Fee Transactions

Gas abstraction is arguably the most transformative feature for user onboarding.[41] It tackles the inflexible gas payment model of EOAs head-on through the use of Paymasters.[36]

- **Gasless Transactions:** DApps can choose to sponsor transaction fees for their users.[15] This "gasless" model removes the initial hurdle of requiring a new user to acquire a network's native token before they can perform their first action.[37] For example, a blockchain game could sponsor the transaction for a user to mint their first in-game item, or an NFT marketplace could sponsor the first listing. This is achieved through a process involving "meta-transactions," where the user signs the transaction's intent off-chain, and a relayer (in ERC-4337, the Bundler and Paymaster system) submits it on-chain and pays the fee.[42]
- **ERC-20 Fee Payments:** Instead of full sponsorship, a Paymaster can be configured to allow users to pay for gas using stablecoins (like USDC or USDT) or other ERC-20 tokens they already hold.[14] The Paymaster handles the complexity of swapping these tokens for the required native token to pay the Bundler, providing a much more convenient experience for users who primarily transact in non-native assets.[38]

Transaction Batching: Simplifying Complex dApp Interactions

Smart accounts can execute multiple distinct operations as a single, atomic on-chain transaction.[2] This feature, often called "transaction batching" or "multicall," has profound implications for UX and efficiency.[16]

- **Improved User Flow:** Instead of prompting the user for multiple signatures for a multi-step process (e.g., approve a token, then swap it), a dApp can bundle these actions into one UserOperation. The user signs only once, and the entire sequence either succeeds or fails together.[12]
- **Reduced Costs and Time:** By combining operations, users save on the fixed gas cost associated with each individual transaction and reduce the total time spent waiting for

confirmations.[45]

- **Enabling Complex Strategies:** Batching makes sophisticated DeFi interactions feasible in a single click. For instance, a user could execute a flow that involves withdrawing funds from a yield vault, approving a token for a swap on a DEX, and executing that swap, all within one atomic transaction that would have required three separate EOA transactions.[44]

Session Keys: Enabling Seamless, "Invisible" Blockchain Experiences

For dApps that require frequent, low-risk interactions, such as blockchain games or social media platforms, the need to sign every single transaction is a major UX killer. Session keys solve this by creating temporary, restricted sub-keys.[47]

- **Temporary, Scoped Permissions:** A user can authorize a "session key"—a new, temporary keypair often stored in the browser's local storage—to sign transactions on their behalf under strictly defined conditions.[48]
- **Granular Control:** These permissions can be highly granular, limited by a specific time duration (e.g., "this key is valid for the next hour"), a spending limit (e.g., "can spend no more than 50 USDC"), a list of approved contracts to interact with, or even specific functions within those contracts.[49]
- **Seamless Interaction:** Once a session key is authorized, the dApp can use it to submit UserOperations in the background without prompting the user for further signatures. A gamer could perform dozens of in-game actions like crafting items or attacking monsters, and each action would be a seamless on-chain transaction. If the session key is ever compromised, the attacker's capabilities are strictly limited by the pre-defined permissions, and the user's main account remains secure.[47]

3.2. Fortifying Security and Control

Beyond UX enhancements, smart accounts introduce a new dimension of programmable security, allowing users and organizations to enforce rules at the account level.

Multi-Signature (Multisig) Configurations

The ability to require M-of-N signatures to authorize a transaction is one of the most powerful security features of smart accounts.[2] By distributing control among multiple keys (held by different individuals or stored on different devices), it eliminates the single point of failure inherent in the EOA model. This is the cornerstone of security for DAOs, corporate treasuries, and any group managing collective assets, as it prevents a single rogue actor or a single compromised key from draining funds.[51]

Social Recovery: A Decentralized Safety Net

Social recovery is a revolutionary feature that directly addresses the catastrophic risk of losing a private key.[2]

- **The Process:** A user designates a set of "guardians"—these can be trusted friends, family members, other hardware wallets they own, or even third-party services.[53]
- **Recovery Mechanism:** If the user loses access to their primary signing device, they can initiate a recovery process. By contacting a required threshold of their guardians (e.g., 3 out of 5), they can have them sign a special transaction. This transaction doesn't give the guardians access to the funds; it simply authorizes the user's smart account to register a new primary signing key.[55]
- **Trustless Security:** The guardians never have direct control over the assets. Their only power is to collectively approve the replacement of the owner's key.[54] This mechanism, first proposed by Vitalik Buterin, makes self-custody a viable and far less intimidating option for the average person, as it provides a decentralized alternative to the "all-or-nothing" nature of seed phrases.[56]

Programmable Security: Spending Limits, Whitelists, and More

The programmable nature of smart accounts allows for an almost infinite variety of custom security policies to be embedded directly into the wallet's logic.[57]

- **Spending Limits:** An account can be programmed with a daily or weekly withdrawal limit. Any transaction exceeding this limit would require additional authorization, such as a multisig approval or a time delay.[14]
- **Address Whitelisting/Allowlisting:** Users can configure their account to only permit transfers to a pre-approved list of addresses, effectively preventing funds from being sent to a malicious or unknown address even if a key is compromised.[14]
- **Account Freezing and Time-locks:** In case of a suspected security breach, a user could

trigger an account freeze, temporarily halting all outgoing transactions. Similarly, large transfers could be subjected to a mandatory time-lock, giving the owner a window to cancel the transaction if it was unauthorized.[14]

These features collectively transform a blockchain account from a simple, passive container of assets into an active, intelligent agent that can enforce its owner's security preferences on-chain.

Section 4: Case Study: Safe — The Industry Standard for Smart Accounts

While ERC-4337 provides the theoretical blueprint for account abstraction, **Safe** (formerly Gnosis Safe) stands as its most prominent and battle-tested real-world implementation. Long before the finalization of ERC-4337, Safe pioneered the concept of smart contract wallets, establishing itself as the de facto standard for securing high-value digital assets. Today, it is the most trusted decentralized custody protocol, securing over $100 billion in assets for DAOs, enterprises, and individuals, including prominent figures like Vitalik Buterin.[61] An analysis of Safe's architecture and ecosystem reveals a masterclass in balancing security, flexibility, and extensibility.

4.1. An Architectural Deep Dive

Safe's success is built upon a robust and gas-efficient architecture designed with security as its foremost priority.[64]

The Singleton and Proxy Pattern

At the core of Safe's design is the use of a singleton and proxy pattern, which is crucial for both cost efficiency and security.[66]

- **Singleton (Mastercopy):** Instead of deploying a full, complex smart contract for every new user, Safe utilizes a single, heavily audited, and formally verified "Singleton" or "Mastercopy" contract. This one contract contains all the core logic for Safe's functionality, including multi-signature validation, transaction execution, and state management.[64]

- **Proxy Contracts:** Each user's individual Safe account is a very lightweight "Proxy" contract. This proxy contains minimal code and holds the unique state for that account (such as the list of owners and the signature threshold). All function calls made to the proxy are delegated to the central Singleton contract for execution.[65]
- **Benefits:** This pattern provides two major advantages. First, it dramatically reduces the gas cost of creating a new Safe, as deploying a small proxy is much cheaper than deploying the full contract logic each time.[66] Second, it allows for efficient and secure system-wide upgrades. If a new feature or security patch is needed, only the central Singleton contract needs to be updated, and all existing proxy contracts can immediately point to the new version without requiring any action from users.[68]
- **Deterministic Deployment:** Safe uses a SafeSingletonFactory contract with the CREATE2 opcode to deploy these proxies. This allows for the deterministic calculation of a Safe's address before it is ever deployed on-chain, a feature essential for seamless dApp integrations and counterfactual account creation.[65]

Extending Functionality: The Role of Modules and Guards

Safe's architecture deliberately separates its core, high-trust logic (multisig) from more complex, extensible features. This is achieved through a powerful plugin system of Modules and Guards, which allows for customization without compromising the integrity of the core contract.[64]

- **Safe Modules:** Modules are separate, external smart contracts that can be granted special permissions by the Safe owners to execute transactions on its behalf, often bypassing the standard M-of-N signature requirement.[65] This enables powerful automation and custom workflows. For example:
  - The **Allowance Module** allows owners to grant limited spending permissions to another account, creating daily or monthly budgets.[65]
  - A **Social Recovery Module** can be set up to allow guardians to recover access to the Safe.[71]
  - The official **4337 Module** makes a Safe account compatible with the ERC-4337 infrastructure, allowing it to send and receive UserOperations.[63]

    It is critical to understand that enabling a module is a significant security decision. A malicious or poorly coded module could potentially drain all assets from the Safe, as it is granted execution privileges. Therefore, users are strongly advised to only add trusted and thoroughly audited modules.71

- **Safe Guards:** Guards are another type of external smart contract that can be attached to a Safe to add extra validation checks to transactions.[64] Unlike modules, guards cannot initiate transactions themselves. Instead, they act as a "transaction firewall." Before and after every transaction executed by the Safe, a check is performed on the Guard contract. The Guard can be programmed to enforce specific rules, such as only allowing interactions with a whitelist of approved dApp contracts or preventing transactions that would reduce the Safe's balance of a particular token below a certain threshold. If the Guard's checks fail, the entire transaction reverts.[68]

This modular design is a key reason for Safe's trusted status. It allows the core contract to remain minimal and secure, while pushing the complexity and associated risk of new features to optional, opt-in plugins. This creates clear security boundaries and allows for a vibrant ecosystem of third-party extensions to be built on top of the Safe protocol.

4.2. Core Functionality and Ecosystem

While architecturally elegant, Safe's dominance comes from its practical application as the premier platform for collective asset management and its deep integration across the Web3 ecosystem.

The Premier Choice for Multi-Signature Wallets

Safe's foundational feature and primary use case is its customizable multi-signature functionality.[65] Organizations can configure their wallets with any M-of-N signature requirement (e.g., 3-of-5, 4-of-7), ensuring that no single individual has unilateral control over funds.[66] This has made it the indispensable tool for:

- **DAOs (Decentralized Autonomous Organizations):** DAOs use Safe to manage their community treasuries in a transparent and decentralized manner. All spending proposals must be approved on-chain by a quorum of elected signatories, providing a verifiable and trust-minimized system for financial governance.[51]
- **Crypto-Native Companies and Teams:** Startups and investment funds in the Web3 space use Safe to secure their operational funds and investor capital. It enforces internal controls, mitigates the risk of employee theft or error, and provides a clear audit trail of all transactions.[62]
- **Security-Conscious Individuals:** High-net-worth individuals use Safe for their personal holdings, setting up a multisig configuration with keys distributed across multiple

hardware wallets and geographic locations to create a robust defense against theft and loss.[62]

The Safe{Core} SDK and dApp Integration Landscape

To foster a rich ecosystem, Safe provides the **Safe{Core} SDK**, a comprehensive suite of tools that allows developers to seamlessly integrate Safe's smart account functionality into their applications.[75] The SDK is broken down into several modular "kits" [77]:

- **Protocol Kit:** Provides the core functionality for creating and interacting with Safe contracts, such as proposing and executing transactions.[75]
- **API Kit:** Allows developers to interact with the Safe Transaction Service, a backend service that facilitates the off-chain sharing and collection of signatures among owners before a transaction is submitted on-chain.[75]
- **Relay Kit:** Enables gas abstraction features, integrating with relayers and ERC-4337 infrastructure to allow for sponsored transactions or fee payments in ERC-20 tokens.[75]

Thanks to these developer-friendly tools, a vast ecosystem of over 200 dApps has integrated Safe.[63] This allows Safe users to interact directly with leading DeFi protocols like Aave and Uniswap, stake assets, participate in governance, and manage their digital identity without ever leaving the secure environment of the Safe interface.[80] This deep integration has solidified Safe's position not just as a wallet, but as a comprehensive platform for on-chain asset management.

Section 5: The Smart Account Ecosystem: Wallets, Infrastructure, and Adoption

The emergence of ERC-4337 has catalyzed a vibrant and rapidly expanding ecosystem around smart accounts. This landscape is composed of two distinct but deeply interconnected layers: a user-facing "Application Layer" of wallets and dApps, and a developer-facing "Infrastructure Layer" of services that power them. The success of the application layer is entirely dependent on the reliability, performance, and cost-effectiveness of the infrastructure layer, creating a new and dynamic B2B market within Web3.

5.1. The Wallet Landscape: A Comparative Analysis

While Safe dominates the high-value treasury management space, a new generation of smart contract wallets has emerged, focusing on bringing the benefits of account abstraction to everyday users, often with a strong emphasis on mobile-first experiences and simplified onboarding.

- **Argent:** A pioneer in the space, Argent is a mobile-first smart contract wallet that targets newcomers to crypto by focusing on a sleek user experience and enhanced security.[82] It has been a strong proponent of Layer 2 solutions, with deployments on zkSync and Starknet, where lower transaction costs make advanced features more viable. Its key offerings include social recovery, multisig capabilities, and transaction batching (multicall), all designed to make self-custody less intimidating.[82]
- **Ambire:** Ambire positions itself as a versatile, multichain smart account wallet available on both web and mobile platforms. It caters to both new users and DeFi power users by combining a simple interface with advanced features. Its standout innovations include an email and password sign-up option that abstracts away seed phrases for a Web2-like onboarding experience, and a "Gas Tank" feature that allows users to prepay for gas and save on fees across multiple transactions and chains.[82]
- **Other Notable Wallets:** The ecosystem is rich with innovation. Wallets like **UniPass** and **Soul Wallet** are built from the ground up on ERC-4337, while others like **Candide** focus on providing open-source, community-driven solutions.[83] This diversity showcases a healthy and competitive market where different wallets are optimizing for different user segments and use cases.

Table 3: Comparative Analysis of Leading Smart Contract Wallets

Wallet	Primary Use Case	Key Features	Supported Networks	Platform
Safe	Treasury & DAO Management, High-Value Self-Custody [62]	Advanced Multi-Signature, Modular Architecture (Modules/Guards), dApp	15+ EVM Chains (Ethereum, Polygon, Arbitrum, etc.) [63]	Web & Mobile

		Ecosystem [65]		
Argent	L2 DeFi for Newcomers, Mobile-First UX [82]	Social Recovery, Multi-Signature, Layer 2 Focus, Multicall [82]	Ethereum, zkSync, Starknet [82]	Mobile-First
Ambire	Versatile DeFi Power-User, Simplified Onboarding [82]	Gas Tank (Fee Savings), Email/Password Login, Built-in DeFi Integrations [82]	Multiple EVM Chains [82]	Web & Mobile

5.2. Infrastructure Providers: The Backbone of ERC-4337

The functionality of ERC-4337 smart accounts is not magic; it relies on a robust backend infrastructure of Bundlers to process UserOperations and Paymasters to handle gas abstraction. Building and maintaining this infrastructure is complex and resource-intensive, which has led to the rise of specialized providers that offer these components "as-a-service" to dApp and wallet developers.[85]

- **Alchemy:** A leading Web3 development platform, Alchemy provides a comprehensive suite of account abstraction tools. This includes a high-performance, open-source Bundler written in Rust, and the "Gas Manager" API, a powerful Paymaster service that allows developers to easily sponsor transactions and configure gas policies for their users.[85]
- **Stackup:** As a dedicated account abstraction infrastructure company, Stackup is known for its enterprise-grade Bundlers and Paymaster APIs. They focus on providing highly reliable and scalable infrastructure for businesses building on ERC-4337, emphasizing uptime and performance.[85]
- **Pimlico:** Pimlico has quickly become a key infrastructure provider in the ecosystem, offering a high-performance Bundler and a variety of Paymaster solutions (including verifying and ERC-20 paymasters). Their services are used by a significant number of smart contract wallets to power their ERC-4337 capabilities.[85]
- **Biconomy:** Biconomy offers a full "Account Abstraction toolkit" that bundles modular smart accounts, Paymasters, and Bundlers into a single, integrated platform. Their goal is to provide developers with an end-to-end solution for building dApps with a simplified

user experience.[85]

These providers form the critical infrastructure layer that enables the application layer to thrive. They compete on reliability, speed, cost, and the ease of integration of their APIs and SDKs, driving innovation and making account abstraction accessible to a wider range of developers.

5.3. Market Adoption and Growth Metrics

On-chain data provides a clear and compelling picture of accelerating smart account adoption, particularly since the deployment of ERC-4337.

- **Account Creation and Transaction Volume:** The growth has been exponential. As of the fourth quarter of 2024, the total number of deployed ERC-4337 accounts surpassed 1.8 million, with over 960,000 of those created in Q4 alone. The volume of UserOperations processed in Q4 2024 exceeded 5.4 million, representing a 194% increase over the previous quarter.[89]
- **Network Dominance and the Role of L2s:** Layer 2 scaling solutions are the primary venue for smart account activity, as their lower gas costs make the overhead of ERC-4337 more manageable.[90]
  Polygon has emerged as the dominant network, hosting 92% of all monthly active smart accounts. This dominance is largely attributed to successful user acquisition campaigns by dApps on the network that leverage smart account features.[89]
- **Paymaster Usage as a Key Indicator:** The demand for gas abstraction is undeniable. A staggering **97-99%** of all UserOperations utilize a Paymaster to handle gas fees.[89] This indicates that sponsoring transactions or allowing users to pay with stablecoins is not just a niche feature but the primary value proposition driving adoption. In December 2024 alone, over $1 million in gas fees were sponsored through Paymasters, with the majority of this volume being processed by the leading infrastructure providers like Pimlico, Stackup, and Alchemy.[89]
- **dApps Driving Adoption:** The initial wave of smart account adoption is being driven by consumer-facing applications that benefit most from a simplified UX. The leading dApps by UserOperation volume include:
  - **Grindery:** A Telegram bot that allows users to send and receive crypto, which ran a successful token incentive program to drive sign-ups.[89]
  - **FanTV:** A video streaming platform that rewards users with tokens, using smart accounts to streamline the claiming process.[89]
  - **CyberConnect:** A Web3 social network where every user profile is a smart account.[91]

These examples show that the most successful early adopters are using smart accounts to remove friction from high-frequency, low-value interactions, a pattern likely to continue as the ecosystem matures.

Section 6: Challenges, Risks, and the Future of Account Abstraction

Despite its immense promise and rapid growth, the smart account paradigm is not without its challenges. The current implementation via ERC-4337 introduces new complexities, security considerations, and potential centralization risks that must be carefully managed. Simultaneously, the ecosystem is already looking ahead, with new standards and native protocol-level implementations paving the way for the next generation of account abstraction.

6.1. Navigating the Hurdles: Costs, Complexity, and Security

The Gas Overhead of ERC-4337 vs. EOAs

A significant and often-cited challenge of ERC-4337 is its increased gas cost compared to standard EOA transactions.[93] This overhead stems from several factors inherent in its design:

- **On-Chain Validation:** Every UserOperation requires an on-chain call to the validateUserOp function of the smart account, which consumes gas.[94]
- **EntryPoint Logic:** The EntryPoint contract itself performs complex logic for looping through bundles, managing stakes, and paying out fees, all of which adds to the total gas cost.[95]
- **Contract Deployment:** The first transaction for any new smart account must also pay the one-time cost of deploying its contract code.[96]

Empirical analysis has shown that a basic token transfer from an ERC-4337 account can consume approximately 92,901 gas, roughly four times the 21,000 gas required for a standard EOA transfer.[94] This higher baseline cost is a significant trade-off for the flexibility and features offered. However, this comparison is nuanced. For complex dApp interactions that would require multiple EOA transactions, the ability of a smart account to

batch these operations into a single transaction can result in a net *reduction* in total gas fees, in addition to the superior user experience.[14]

Common Security Vulnerabilities and Mitigation Strategies

Moving account logic from the protocol layer into user-deployed smart contracts introduces new attack surfaces that developers must be vigilant about.[97] While the ERC-4337 standard itself is heavily audited, the custom smart accounts, modules, and paymasters built on top of it are all potential sources of vulnerabilities.[98]

- **Smart Contract Bugs:** Standard Solidity vulnerabilities like **reentrancy attacks** (especially in Paymaster contracts that handle token transfers), **oracle manipulation**, and **insecure randomness** can have catastrophic consequences for a smart account.[98]
- **Upgradability Risks:** Many smart accounts are designed to be upgradable to allow for future feature additions. If not managed carefully, this can lead to critical bugs such as **storage collisions**, where an upgrade inadvertently overwrites critical state variables, potentially locking funds or compromising security.[98]
- **Module and Plugin Risk:** As seen with Safe, enabling external modules grants them significant power over the account. A malicious or vulnerable module could be exploited to drain the account's funds entirely. A real-world example of this was a vulnerability discovered in the UniPass wallet, where a combination of flaws could have allowed an attacker to take full control of a user's account by replacing its trusted EntryPoint.[101]

Mitigating these risks requires adherence to security best practices, including following the checks-effects-interactions pattern, using trusted libraries like OpenZeppelin, and, most importantly, undergoing **rigorous, professional security audits** for all smart contract code before deployment.[97]

The Centralization Dilemma: Analyzing Bundlers and Paymasters

While ERC-4337 is designed as a decentralized protocol, there is a tangible risk that its key infrastructure components—Bundlers and Paymasters—could become centralized around a few dominant players.[96] The technical complexity and capital requirements (e.g., holding ETH across multiple chains for gas payments) create economies of scale that favor large, well-funded operators.[103]

- **Potential Consequences:** This centralization could lead to several negative outcomes. A small number of dominant Bundlers could potentially **censor** UserOperations by refusing to include them in bundles. They could also become a single point of failure; if a major Bundler service goes down, it could disrupt a large portion of the smart account ecosystem. Furthermore, centralized Bundlers are in a prime position to extract Maximal Extractable Value (MEV) by front-running transactions within the bundles they create.[103]

- **Mitigation Strategies:** The ERC-4337 design includes several features to counteract these risks. The alt mempool is permissionless, meaning in theory, anyone can run a Bundler and compete, preventing total capture.[21] The development of shared mempool standards and decentralized Bundler networks aims to foster a more resilient and censorship-resistant ecosystem.[104] For Paymasters, the risk is less about censorship and more about reliance on a single service, but the open nature of the standard allows dApps to switch between Paymaster providers or run their own, promoting competition.[39]

6.2. The Road Ahead: Evolving Standards and Native Implementations

The evolution of account abstraction is progressing rapidly, following a classic technology adoption pattern: from a clever but complex add-on (ERC-4337), to a bridging technology for legacy users (EIP-7702), and ultimately toward native integration into the core protocol itself.

Beyond ERC-4337: Evolving Ethereum Standards

- **ERC-6900: Modular Smart Accounts:** This proposal seeks to standardize the *internal* architecture of smart accounts by creating a universal plugin system.[106] Instead of monolithic wallet designs, an ERC-6900 account would be a minimal base that can have standardized modules for validation (e.g., passkey, multisig), execution, and hooks (e.g., spending limits) installed, removed, and upgraded. This would foster a more open and interoperable ecosystem where users could mix and match plugins from different developers, much like installing apps on a smartphone, preventing vendor lock-in.[107]
- **EIP-7702: Upgrading Existing EOAs:** Set to be included in Ethereum's upcoming "Pectra" hard fork, EIP-7702 is a groundbreaking proposal that addresses the biggest adoption hurdle of ERC-4337: the need for users to migrate from their existing EOAs.[109] It introduces a new transaction type that allows an EOA to **temporarily delegate its authority to a smart contract for the duration of a single transaction**.[111] This means a user with a standard MetaMask wallet could, for instance, execute a batched transaction or use a Paymaster without having to create a new smart account address or move their funds. EIP-7702 is designed to be fully compatible with

the existing ERC-4337 infrastructure and is seen as a complementary "bridge" to bring the benefits of account abstraction to Ethereum's massive existing user base.[109]

Native Implementations on L2s: A Look at StarkNet and zkSync

Newer Layer 2 networks, unburdened by Ethereum's legacy, are building account abstraction directly into their core protocols, offering a glimpse into its long-term future.

- **Starknet:** On Starknet, account abstraction is native and absolute. **There are no EOAs; every account is a smart contract by default**.[114] This deep integration allows for more advanced features, such as supporting different cryptographic signature schemes (like those used by smartphone secure enclaves) at the protocol level. It also eliminates the need for a separate Bundler and EntryPoint system, as transaction validation and execution are handled directly by the network's Sequencer, resulting in a cleaner and potentially more efficient architecture.[114]
- **zkSync:** zkSync also implements native account abstraction at the protocol level, but in a way that is highly inspired by ERC-4337.[117] It unifies the transaction mempool, allowing both EOAs and smart accounts to coexist seamlessly and, crucially, enabling even traditional EOAs to benefit from Paymasters.[119] This provides a more integrated and efficient experience compared to Ethereum's layered approach, while still maintaining some familiarity for existing users.[117]

Table 4: Native AA vs. ERC-4337 - A Technical Comparison

Aspect	ERC-4337 (Ethereum)	Native AA (Starknet / zkSync)
Implementation Level	Application Layer (Smart Contracts on top of the protocol) [24]	Protocol Layer (Integrated into the blockchain's core logic) [114]
Account Types	Coexistence of EOAs and Smart Contract Accounts (SCAs) [22]	Only Smart Accounts (Starknet); Unified accounts (zkSync) [115]

Transaction Flow	Dual Mempool: Standard transactions for EOAs, alternate mempool for UserOperations [21]	Unified Mempool and transaction flow for all account types [119]
Infrastructure	Requires an external, decentralized network of Bundlers and a global EntryPoint contract [24]	Integrated directly into the network's core operators (Sequencer/Proposer) [116]
Gas Overhead	Higher baseline cost due to on-chain validation and EntryPoint logic [93]	Lower overhead as validation logic is part of the core protocol, not an extra contract call [121]

Vitalik Buterin's Vision for the Future of Ethereum Accounts

Synthesizing Vitalik Buterin's public statements reveals a clear long-term vision where smart accounts become the dominant paradigm on Ethereum.[122] The "endgame" is an ecosystem where the default user experience is a smart contract wallet equipped with robust security features like multi-signature control and social recovery. He views pragmatic proposals like EIP-7702 not as a final destination but as crucial stepping stones that help bridge the gap for existing users.[110] Furthermore, his vision extends to leveraging the cryptographic agility of smart accounts to future-proof the network against long-term threats like quantum computing, by enabling users to easily upgrade their accounts to use quantum-resistant signature schemes.[124] This forward-looking perspective underscores that account abstraction is not just about improving today's UX, but about building a more resilient and adaptable foundation for the future of digital ownership.

Conclusion

The transition from Externally Owned Accounts to Smart Accounts represents a fundamental and necessary evolution for the blockchain ecosystem. Driven by the principle of Account Abstraction and standardized by ERC-4337, this new paradigm directly addresses the critical barriers of poor user experience and rigid security that have long hindered mainstream adoption.

By making accounts programmable, smart accounts unlock a suite of transformative features. Gas abstraction, transaction batching, and session keys work in concert to create seamless, intuitive interactions that begin to rival the polish of modern Web2 applications. Simultaneously, features like multi-signature security, social recovery, and customizable spending limits provide a level of safety and control that makes self-custody a viable and far less intimidating prospect for the average user.

The ecosystem has responded with vigor. A thriving landscape of innovative wallets like Safe, Argent, and Ambire now offers users a diverse range of choices, while a robust infrastructure layer powered by providers such as Alchemy, Stackup, and Pimlico ensures the reliable processing of smart account transactions. On-chain data confirms this momentum, with exponential growth in account creation and transaction volume, particularly on cost-effective Layer 2 networks.

However, the path forward is not without its challenges. The increased gas overhead of ERC-4337, the new security vulnerabilities introduced by programmable logic, and the potential risks of infrastructure centralization are significant hurdles that the community must continue to address through rigorous auditing, optimization, and the promotion of decentralized alternatives.

The future of account abstraction is bright and dynamic. Evolving standards like ERC-6900 promise a more modular and interoperable future for wallet design, while the landmark EIP-7702 is set to bridge the gap for millions of existing EOA users. Meanwhile, native implementations on networks like Starknet and zkSync offer a compelling glimpse into the "endgame": a future where programmable, secure, and user-friendly accounts are not an add-on, but the native fabric of the blockchain itself. This ongoing revolution is more than a technical upgrade; it is the critical work of building a Web3 that is finally ready for everyone.

Works cited

1. What are Ethereum Accounts? | QuickNode Guides, accessed September 29, 2025, https://www.quicknode.com/guides/ethereum-development/getting-started/what-are-ethereum-accounts
2. What is a smart account? | MetaMask Help Center, accessed September 29, 2025, https://support.metamask.io/configure/accounts/what-is-a-smart-account/
3. EOA vs. Smart Contract Account: What Is the Difference? - The Ambire Wallet Blog, accessed September 29, 2025, https://blog.ambire.com/eoas-vs-smart-contract-accounts/
4. Understanding Smart Accounts, accessed September 29, 2025, https://docs.stackup.fi/docs/understanding-smart-accounts
5. What Are Externally Owned Accounts (EOAs) in Ethereum? - Unchained Crypto, accessed September 29, 2025,

https://unchainedcrypto.com/externally-owned-accounts-ethereum/

6. Externally Owned Wallets (EOAs) vs. Smart Contract Wallets vs. Embedded Wallets - Turnkey, accessed September 29, 2025, https://www.turnkey.com/blog/externally-owned-wallets-eoas-vs-smart-contract-wallets-vs-embedded-wallets
7. How do ERC-4337 smart contract wallets work? - Alchemy, accessed September 29, 2025, https://www.alchemy.com/overviews/how-do-smart-contract-wallets-work
8. Understanding Crypto Wallets: EOAs vs Smart Contract Accounts | by Bhaisaaab | Medium, accessed September 29, 2025, https://medium.com/@Bhaisaaab_/understanding-crypto-wallets-eoas-vs-smart-contract-accounts-550d415546ce
9. Embedded EOAs vs. Smart Wallets vs. Smart EOAs - Alchemy, accessed September 29, 2025, https://www.alchemy.com/blog/embedded-eoa-smart-wallets-comparison
10. An ultimate guide to Web3 Wallets: Externally Owned Account and Smart Contract Wallet, accessed September 29, 2025, https://blog.web3auth.io/an-ultimate-guide-to-web3-wallets-externally-owned-account-and-smart-contract-wallet/
11. Why the Future of Ethereum is Smart (Accounts) - Safe, accessed September 29, 2025, https://safe.mirror.xyz/vZHodil1NLJbz4fd0vuil0hyLHNGCBH8oLyMzGTb4sc?collectors=true
12. Account Abstraction Explained: A Developer's Guide - Thirdweb, accessed September 29, 2025, https://thirdweb.com/learn/guides/account-abstraction-the-developers-guide
13. Account Abstraction and ERC-4337 - Part 1 | QuickNode Guides, accessed September 29, 2025, https://www.quicknode.com/guides/ethereum-development/wallets/account-abstraction-and-erc-4337
14. What Are Smart Accounts? - Transak, accessed September 29, 2025, https://transak.com/blog/what-are-smart-accounts
15. What is the Difference between Smart Wallets and Externally Owned Accounts | by ChainEx, accessed September 29, 2025, https://medium.com/@chainexeth/what-is-the-difference-between-smart-wallets-and-externally-owned-accounts-9e4f73a6ac69
16. Account Types - Circle Docs, accessed September 29, 2025, https://developers.circle.com/w3s/programmable-wallets-account-types
17. Introductory Guide to Account Abstraction (ERC-4337) - Blocknative, accessed September 29, 2025, https://www.blocknative.com/blog/account-abstraction-erc-4337-guide
18. The ABCs of ERC-4337: A Beginner's Guide, accessed September 29, 2025, https://www.arianee.com/post/the-abcs-of-erc-4337-a-beginners-guide
19. Smart Account Meaning - Thirdweb, accessed September 29, 2025, https://thirdweb.com/learn/glossary/smart-account

20. An Introduction to ERC-4337 (Account Abstraction) Standard - Unchained Crypto, accessed September 29, 2025, https://unchainedcrypto.com/erc-4337-account-abstraction-standard/
21. ERC 4337, accessed September 29, 2025, https://www.erc4337.io/
22. What is ERC4337? A Complete Guide to Ethereum Account Abstraction - Eco, accessed September 29, 2025, https://eco.com/support/en/articles/11850386-what-is-erc4337-a-complete-guide-to-ethereum-account-abstraction
23. Why ERC-4337 “Account Abstraction” Falls Short of Radix Smart Accounts | The Radix Blog, accessed September 29, 2025, https://www.radixdlt.com/blog/comparing-account-abstraction-and-radix-smart-accounts
24. What is Account Abstraction (ERC-4337)? - Alchemy, accessed September 29, 2025, https://www.alchemy.com/overviews/what-is-account-abstraction
25. Account abstraction: A beginner's guide to Ethereum's ERC-4337 standard - Cointelegraph, accessed September 29, 2025, https://cointelegraph.com/learn/articles/account-abstraction-guide-to-ethereums-erc-4337-standard
26. Account Abstraction - OpenZeppelin Docs, accessed September 29, 2025, https://docs.openzeppelin.com/contracts/5.x/account-abstraction
27. ERC-4337 Decoding EntryPoint and UserOperation - Notion, accessed September 29, 2025, https://biconomy.notion.site/ERC-4337-Decoding-EntryPoint-and-UserOperation-c9589d072041413486d2caef49260f9f
28. Operations | Epoch Protocol Docs, accessed September 29, 2025, https://docs.epochprotocol.xyz/account-abstraction/erc-4337/operations
29. What is ERC-4337? - Stackup, accessed September 29, 2025, https://www.stackup.fi/resources/what-is-eip-4337
30. Intro to Account Abstraction: What is a Bundler? - Alchemy, accessed September 29, 2025, https://www.alchemy.com/overviews/what-is-a-bundler
31. Bundlers - ERC-4337 Documentation, accessed September 29, 2025, https://docs.erc4337.io/bundlers/index.html
32. What is ERC-4337? - Safe Docs, accessed September 29, 2025, https://docs.safe.global/advanced/erc-4337/overview
33. The EntryPoint Contract - ERC-4337 Documentation, accessed September 29, 2025, https://docs.erc4337.io/smart-accounts/entrypoint-explainer.html
34. EntryPoint contract gains temporary control of an ERC-4337 wallet - Block Magnates, accessed September 29, 2025, https://blog.blockmagnates.com/entrypoint-contract-gains-temporary-control-of-an-erc-4337-wallet-c10b3c4aef2e
35. A deep dive into the main components of ERC-4337: Account Abstraction Using Alt Mempool — Part 2 | by Antonio Viggiano | Oak Security | Medium, accessed September 29, 2025, https://medium.com/oak-security/a-deep-dive-into-the-main-components-of-erc-4337-account-abstraction-using-alt-mempool-part-2-0c62617d9ebe

36. Paymasters - OpenZeppelin Docs, accessed September 29, 2025, https://docs.openzeppelin.com/community-contracts/0.0.1/paymasters
37. Gasless Transactions: How You Can Save Thousands in Crypto Fees - Nonbank, accessed September 29, 2025, https://nonbank.io/blog/gasless-transactions-crypto-fees
38. What are Paymasters? (ERC-4337) - Alchemy, accessed September 29, 2025, https://www.alchemy.com/overviews/what-is-a-paymaster
39. Paymasters - ERC-4337 Documentation, accessed September 29, 2025, https://docs.erc4337.io/paymasters/index.html
40. Smart Account Security: Auditing Account Abstraction | by horsefacts | Code4rena - Medium, accessed September 29, 2025, https://medium.com/code4rena/smart-account-security-69b544c0da86
41. Account Abstraction (ERC-4337) - Ministry of Programming, accessed September 29, 2025, https://ministryofprogramming.com/blog/account-abstraction-erc-4337-2
42. Gasless Transactions Explained: How They Work and Are They Safe?, accessed September 29, 2025, https://www.coinsdo.com/en/blog/what-are-gasless-transactions
43. Gasless/Meta Transactions - Venly, accessed September 29, 2025, https://docs.venly.io/docs/what-are-meta-transactions
44. Batch Transactions with the Smart Wallet - thirdweb blog, accessed September 29, 2025, https://blog.thirdweb.com/guides/how-to-batch-transactions-with-the-thirdweb-sdk/
45. Send batch transactions | MetaMask developer documentation, accessed September 29, 2025, https://docs.metamask.io/wallet/how-to/send-transactions/send-batch-transactions/
46. Safe: Smart Accounts - Fluidkey, accessed September 29, 2025, https://www.fluidkey.com/blog/safe-smart-accounts
47. What are Session Keys? The Complete Guide to Building Invisible ..., accessed September 29, 2025, https://blog.thirdweb.com/what-are-session-keys-the-complete-guide-to-building-invisible-blockchain-experiences-with-account-abstraction/
48. How Session Keys on Starknet Revolutionize dApp UX with Gasless Transactions, accessed September 29, 2025, https://www.starknet.io/blog/session-keys-on-starknet-unlocking-gasless-secure-transactions/
49. Session Key Plugin - Web3 Wallet Tools - Alchemy, accessed September 29, 2025, https://www.alchemy.com/dapps/session-key-plugin
50. Transactions - thirdweb docs, accessed September 29, 2025, https://portal.thirdweb.com/engine/v3/guides/session-keys
51. Wallet Security: Best Practices For Keeping Your Crypto Safe - Hacken, accessed September 29, 2025, https://hacken.io/discover/wallet-security/
52. Multisignature crypto wallets are the safest bet for DAOs - Cointelegraph,

accessed September 29, 2025, https://cointelegraph.com/news/multi-signature-crypto-wallets-are-the-safest-bet-for-daos
53. Are you using social recovery for your crypto wallets? : r/CryptoCurrency - Reddit, accessed September 29, 2025, https://www.reddit.com/r/CryptoCurrency/comments/14fz3l9/are_you_using_social_recovery_for_your_crypto/
54. Social Recovery Wallets Explained | What You Need to Know, accessed September 29, 2025, https://moneyzine.com/explore/social-recovery-wallets-explained/
55. Social Recovery - Polkadot Wiki, accessed September 29, 2025, https://wiki.polkadot.com/kusama/kusama-social-recovery/
56. What is a Social Recovery Wallet? - Gate.com, accessed September 29, 2025, https://www.gate.com/learn/articles/what-is-a-social-recovery-wallet/676
57. Programmable Money: CBDCs and the New Era of Policy-Driven Payments, accessed September 29, 2025, https://www.finextra.com/blogposting/29435/programmable-money-cbdcs-and-the-new-era-of-policy-driven-payments
58. What is Programmable Money and Why Does It Matter? - Lightspark, accessed September 29, 2025, https://www.lightspark.com/knowledge/what-is-programmable-money-and-why-it-matters
59. Smart Contract Wallet: 5 Epic Benefits | Braavos WalletBraavos, accessed September 29, 2025, https://braavos.app/smart-contract-wallet/
60. Full Guide to Blockchain Account Abstraction - Cyfrin, accessed September 29, 2025, https://www.cyfrin.io/blog/what-is-blockchain-account-abstraction-a-5-minute-guide
61. Top 9 Multisig Wallets for Crypto in 2025 - Debut Infotech, accessed September 29, 2025, https://www.debutinfotech.com/blog/top-multisig-wallets
62. Safe Wallet Review 2025: Pros, Cons, & Features - Milk Road, accessed September 29, 2025, https://milkroad.com/reviews/safe-wallet/
63. Ethereum Smart Accounts, accessed September 29, 2025, https://safe.global/
64. What is Safe? - Delphi Digital, accessed September 29, 2025, https://members.delphidigital.io/projects/safe
65. How do Safe Smart Accounts work? – Safe Docs, accessed September 29, 2025, https://docs.safe.global/advanced/smart-account-overview
66. Safe (Gnosis Safe) — Smart contract wallets for teams & DAOs | by BizThon - Medium, accessed September 29, 2025, https://medium.com/@BizthonOfficial/safe-gnosis-safe-smart-contract-wallets-for-teams-daos-a48faf7d352e
67. Building a secure Web3 future with Protofire using Safe's Modular Architecture - Medium, accessed September 29, 2025, https://medium.com/protofire-blog/building-a-secure-web3-future-with-protofire-using-safes-modular-architecture-a866cbf7a10d

68. Safe Smart Accounts & Diamond Proxies, accessed September 29, 2025, https://safe.mirror.xyz/P83_rVQuUQJAM-SnMpWvsHIN8oLnCeSncD1txyMDqpE
69. safe-global/safe-smart-account: Safe allows secure management of blockchain assets. - GitHub, accessed September 29, 2025, https://github.com/safe-global/safe-smart-account
70. Safe Modular Smart Account Architecture – Explained, accessed September 29, 2025, https://safe.mirror.xyz/t76RZPgEKdRmWNIbEzi75onWPeZrBrwbLRejuj-iPpQ
71. Safe Modules - Safe Docs, accessed September 29, 2025, https://docs.safe.global/advanced/smart-account-modules
72. A collection of modules that can be used with the Safe contract - GitHub, accessed September 29, 2025, https://github.com/safe-global/safe-modules
73. Safe(Wallet) Multisig Guide For Projects - Bitbond, accessed September 29, 2025, https://www.bitbond.com/resources/gnosis-safe-multisig-guide-for-projects/
74. Multi-Signature Wallets: Secure Your Assets with Multi-Signature Wallets - Bitcoin mining: mine the BTC cryptocurrency | ECOS - Crypto investment platform, accessed September 29, 2025, https://ecos.am/en/blog/multi-signature-wallets-secure-your-assets-with-multi-signature-wallets/
75. Safe{Core} SDK, accessed September 29, 2025, https://docs.safe.global/sdk/overview
76. Safe{Core} — The #1 Smart Account Infrastructure across EVM, accessed September 29, 2025, https://safe.global/core
77. The Safe{Core} SDK allows builders to add account abstraction functionality into their apps. - GitHub, accessed September 29, 2025, https://github.com/safe-global/safe-core-sdk
78. Building Account Abstraction with Safe - Gnosis Chain, accessed September 29, 2025, https://docs.gnosischain.com/technicalguides/account-abstraction/Safe%20and%20supported%20AA%20infra%20providers/
79. Safe Multisig Wallet: Secure Asset Management - Protofire, accessed September 29, 2025, https://protofire.io/projects/gnosis-safe-wallet
80. dApp Store Integration - MSafe, accessed September 29, 2025, https://www.m-safe.io/insights/dapp-store-integration
81. DApp Integration - Blockscout Docs, accessed September 29, 2025, https://docs.blockscout.com/using-blockscout/blockscout-apps/dapp-integration
82. The Best 5 Account Abstraction Wallets in 2025, accessed September 29, 2025, https://blog.ambire.com/best-account-abstraction-wallets/
83. List of 44 Smart Contract Wallets (2025) - Alchemy, accessed September 29, 2025, https://www.alchemy.com/dapps/best/smart-contract-wallets
84. Decoding Safe: Analyzing Multi-Signature Wallets and Exploring DAO Treasury Management | by DAOBase | Medium, accessed September 29, 2025, https://medium.com/@daobase_ai/decoding-safe-analyzing-multi-signature-wallets-and-exploring-dao-treasury-management-64e452a3f13
85. Account Abstraction - Base Documentation, accessed September 29, 2025,

https://docs.base.org/base-chain/tools/account-abstraction
86. Account Abstraction - Portal, accessed September 29, 2025, https://www.portalhq.io/platform/account-abstraction
87. Top 6 Account Abstraction Providers: An In-Depth Review | by Prez Thomas - Medium, accessed September 29, 2025, https://medium.com/coinmonks/top-6-account-abstraction-providers-an-in-depth-review-3a09b9fc707c
88. List of 8 Account Abstraction (ERC-4337) Bundlers (2025) - Alchemy, accessed September 29, 2025, https://www.alchemy.com/dapps/best/account-abstraction-erc-4337-bundlers
89. Smart Accounts Adoption Accelerated in Q4 2023 - Alchemy, accessed September 29, 2025, https://www.alchemy.com/blog/smart-accounts-adoption-accelerated-in-q4-2023
90. State of Wallets - Part 2: Smart Accounts (Account Abstraction – From Theory to Practice), accessed September 29, 2025, https://www.coinbase.com/blog/state-of-wallets-2
91. Smart Accounts Take Off in Q3 (ERC-4337 Statistics) - Alchemy, accessed September 29, 2025, https://www.alchemy.com/blog/erc-4337-statistics-q3-2023
92. Empirical Validation of Account Abstraction: A Quantitative Analysis of Transaction Performance and Adoption Patterns in the Ethereum Blockchain - Mason Publishing Journals, accessed September 29, 2025, https://journals.gmu.edu/jssr/article/view/5234
93. The Risks of Smart Wallets: Navigating Account Abstraction in Web3 | by Dana Love, accessed September 29, 2025, https://medium.com/@DanaFLove/the-risks-of-smart-wallets-navigating-account-abstraction-in-web3-a6c12707cd4f
94. A Measurement Investigation of ERC-4337 Smart Contracts on ..., accessed September 29, 2025, http://www.conf-icnc.org/2024/papers/p1164-lin.pdf
95. Everything we care about Account Abstraction- Ethereum Account Evolution brought by ERC4337 | by Rui Shang | Medium, accessed September 29, 2025, https://medium.com/@poporuii/should-we-bullish-on-account-abstraction-aa-and-how-to-evaluate-erc4337-5f15e30507e
96. Demystifying ERC-4337 & Smart Contract Wallets: 3 Common Misconceptions & 3 Key Challenges | by Paul Huh | Blocto, accessed September 29, 2025, https://medium.portto.com/demystifying-erc-4337-smart-contract-wallets-3-common-misconceptions-3-key-challenges-bd7376dee237
97. Smart Account Security - H-X Technologies, accessed September 29, 2025, https://www.h-x.technology/blog/smart-account-security
98. ERC-4337 Primer: What You Need to Know | Zellic — Research, accessed September 29, 2025, https://www.zellic.io/blog/erc-4337-primer
99. Smart Contract Security: 12 Solidity Vulnerabilities Every Developer Must Know - Medium, accessed September 29, 2025, https://medium.com/@dehvcurtis/smart-contract-security-12-solidity-vulnerabilities-every-developer-must-know-0c1772f61a79

100. A Guide to Smart Contract Security | Hedera, accessed September 29, 2025, https://hedera.com/learning/smart-contracts/smart-contract-security
101. Fireblocks researchers uncover first Account Abstraction wallet vulnerability, accessed September 29, 2025, https://www.fireblocks.com/blog/fireblocks-researchers-uncover-first-account-abstraction-wallet-vulnerability/
102. Account Abstraction and ERC-4337: the End of Seed Phrases and the Start of Mass Web3 Adoption | Technorely, accessed September 29, 2025, https://technorely.com/insights/account-abstraction-and-erc-4337-the-end-of-seed-phrases-and-the-start-of-mass-web-3-adoption
103. Challenges of EIP4337-bundler in Layer2 implementation - HackMD, accessed September 29, 2025, https://hackmd.io/@Jayden-sudo/HJSMHwlI2
104. ERC-4337 Bundlers Comparison, accessed September 29, 2025, https://bundle.rs/
105. ERC-4337 Documentation, accessed September 29, 2025, https://docs.erc4337.io/
106. Inside ERC-6900: Building Modular Smart Accounts on Ethereum | by Anandi Sheladiya, accessed September 29, 2025, https://medium.com/@anandi.sheladiya/inside-erc-6900-building-modular-smart-accounts-on-ethereum-f77bc3ea4a07
107. ERC-6900, accessed September 29, 2025, https://erc6900.io/
108. Explanation of ERC-6900: Modular Smart Contract Accounts and ..., accessed September 29, 2025, https://hackernoon.com/explanation-of-erc-6900-modular-smart-contract-accounts-and-plugins
109. Account abstraction on Ethereum: From ERC-4337 to EIP ... - Turnkey, accessed September 29, 2025, https://www.turnkey.com/blog/account-abstraction-erc-4337-eip-7702
110. Vitalik rallies support for temporary smart wallets on Ethereum - Blockworks, accessed September 29, 2025, https://blockworks.co/news/vitalik-buterin-eip-account-abstraction
111. What is EIP-7702? - Safe Docs, accessed September 29, 2025, https://docs.safe.global/advanced/eip-7702/overview
112. EIP-7702 Explained: How it Works and Everything You Need to Know - Web3Auth Blog, accessed September 29, 2025, https://blog.web3auth.io/eip-7702-explained-how-it-works-and-everything-you-need-to-know/
113. In-Depth Discussion on EIP-7702 and Best Practices | by SlowMist | Medium, accessed September 29, 2025, https://slowmist.medium.com/in-depth-discussion-on-eip-7702-and-best-practices-968b6f57c0d5
114. Account Abstraction: Your Crypto Advantage - Starknet, accessed September 29, 2025, https://www.starknet.io/blog/account-abstraction/
115. Accounts - OpenZeppelin Docs, accessed September 29, 2025, https://docs.openzeppelin.com/contracts-cairo/0.14.0/accounts

116. StarkNet Account Abstraction Model - Part 1, accessed September 29, 2025, https://community.starknet.io/t/starknet-account-abstraction-model-part-1/781
117. Introduction - ZKsync Docs, accessed September 29, 2025, https://docs.zksync.io/zksync-protocol/account-abstraction
118. Native Account Abstraction (AA) with zkSync Era | LearnWeb3, accessed September 29, 2025, https://learnweb3.io/lessons/native-account-abstraction-aa-with-zk-sync-era/
119. Native AA vs EIP 4337 - ZKsync Docs, accessed September 29, 2025, https://docs.zksync.io/zksync-protocol/differences/native-vs-eip4337
120. Design - ZKsync Docs, accessed September 29, 2025, https://docs.zksync.io/zksync-protocol/account-abstraction/design
121. Account Abstraction Gas Fees Explained: Paymasters, Bundlers, and Cost Optimization, accessed September 29, 2025, https://blog.thirdweb.com/account-abstraction-gas-fees-paymasters-bundlers-cost-optimization/
122. Cryptocurrency enters its "adolescent stage": What is the future ..., accessed September 29, 2025, https://www.bitget.com/news/detail/12560604991392
123. Vitalik mentioned the account abstraction revolution again: | Web3狼叔 on Binance Square, accessed September 29, 2025, https://www.binance.com/en/square/post/23519905830226
124. Vitalik Buterin Outlines Ethereum's Decade-Long Vision - ForkLog, accessed September 29, 2025, https://forklog.com/en/vitalik-buterin-outlines-ethereums-decade-long-vision/