This is a part of a longer research on the topic which ultimately intends to establish a common framework to originate on-chain data and process it to the networks supporting data minimization to balance user privacy and compliance needs.

I wanted to thank Erwin Voloder, Pete Cooling, Kim Hamilton and Martin Schaffner, who personally supported this research in different ways and the people from ERC-4337 Team, specifically @yoavw , @zincoshine AA Mafia Telegram Group, Alex and Eric for further support. This is an ongoing project, I hope you appreciate the forward thinking and we are continuing to work on it. Any feedback or support is appreciated, in light blue hyperlinked public content for reference.

ABSTRACT

This research explores the innovative integration of Self-Sovereign Identity (SSI) systems with Account Abstraction (AA) to enhance privacy, compliance, and user experience on the Ethereum blockchain. By leveraging verifiable credentials (VCs) and Ethereum attestations (EAs), the proposed framework empowers bundlers to pre-validate transactions, reducing on-chain reverts and penalties. The implementation of Merkle proofs and Zero-Knowledge Proofs (ZKPs) ensures efficient data processing and minimal disclosure, addressing both scalability and security concerns.

A significant highlight is the introduction of keystore rollups, which manage encryption keys off-chain to facilitate atomic cross-chain transactions. These rollups promote interoperability between sequencing networks, ensuring seamless data exchange and robust security. The research underscores the potential of modular blockchain networks and advocates for the adoption of ERC-4337 and future cryptographic advancements to standardize and optimize Ethereum's execution layer.

[

1223×725 19.3 KB

](https://ethresear.ch/uploads/default/original/3X/2/e/2e77cc94de2442ed8b9e865737b216b820de2396.png)

THE WORK

Using verifiable credentials (VCs) or Ethereum attestations (EAs) to empower bundlers in a shared mempool can potentially reduce the risk of on-chain reverts by sequencers and penalties for block builders. By including data assertions in user operations, bundlers are supported in the pre-validate certain aspects of the transactions. This pre-validation can help identify and filter out operations with a high risk of reverting during execution. This reduces the chances of sequencers including such operations in blocks, lowering the overall revert risk. VCs and EAs provide cryptographic proofs that can be verified by sequencers. These proofs can increase trust in the validity of the operations, making sequencers more confident about including them in blocks.

This has potential positive consequences in terms of reducing risk of penalties for block builders. . Data assertions containing proofs about transaction ordering can help sequencers resist such manipulation attempts, potentially reducing penalties for block builders who are susceptible to MEV.

The effectiveness of data assertions in reducing reverts and penalties depends on factors like network assertation standardization and native account abstraction adoption but using VCs and EAs in a shared mempool with ERC-4337 could be considered as first step to it and has the potential to significantly reduce the risk of on-chain reverts and penalties for sequencers and block builders.

Overall, the complexity of implementing and maintaining new verification logic for bundlers may be compensated by an overall better synergy with the other actors in the network. Specifically on the logic side, general standards like did:pkh or did:dis could find common ground of assessment. In fact, the same verification logic is standards practice for user authentication where user store different assertation on the identity within the wallet application. Extending the same logic approach into the context of smart contract accounts we would suggest for bundlers to store data processing assertion into the mempool of future transactions until consolidated finality is reached.

At the same time, a standardized method for signature validation will be required. ERC-1271 enhances the security and interoperability of the system. So to ensure that all signatures whether aggregated or individual, are verified uniformly, reducing the risk of fraud and enhancing trust in the system.

In that context ERC-1271 facilitates efficient cross-chain operations by ensuring that signatures required for these transactions are validated consistently across different networks. This reduces the complexity and potential errors in verifying multiple signatures, making cross-chain transactions more efficient and reliable.

The same verification and validation logic could be extended into a future unified mempool once potentially account abstraction would be natively implemented following 7560/7562 rules.

Including proofs of VCs or EAs adds data to user operations, potentially leading to larger bundles and higher gas costs. This can put pressure on mempool capacity. Utilizing Merkle proofs within VCs can allow bundlers to verify the inclusion of specific data elements without needing the entire credential, reducing the amount of data processed. Merkle trees are a cryptographic data structure that allows for efficient verification of the membership of a specific data element within a larger set.

After user authentication, for each user operation, the bundler includes a Merkle proof. This proof is a concise piece of data that demonstrates the specific assertion's inclusion within the Merkle tree of the entire bundle. This can significantly reducing the overall size of the bundle compared to including the full VCs or EAs.

VCs and EAs still remain crucial for containing the actual data and claims associated with user operations. Merkle proof act as a verification tool. They provide a concise way to prove that a specific VC or EA exists within a larger bundle of assertions without revealing the entire content of the VC/EA itself. This may help mitigate pressure on the mempool and potentially lowers gas costs.

This would imply for bundlers to access a privacy preserving shared mempool ID and establish a dedicated membership set for associated bundle of transactions, so that Sequencers can verify the inclusion of user operations in the bundle. Sequencers or other network participants can efficiently verify the validity of data assertions using the Merkle proofs. They only need to verify the proof itself, not the entire data within the assertion, reducing the computational burden.

VCs and EAs rely on the underlying logic verification process for authentication and authorization that should be implemented by bundlers. Verifying VCs or EAs requires processing cryptographic proofs. This can be computationally expensive for bundlers, especially for complex assertions.

Merkle proofs can be combined with ZKPs for a powerful solution.ZKPs can be used within the data assertions themselves to prove the validity of claims without revealing the underlying data. Merkle proofs ensure the integrity of the bundle and verify the inclusion of specific assertions with ZKPs within the bundle. Also, using ZKPs in VCs or EAs could reduce the amount of data included in the bundle, minimizing disclosure, and lowering computational costs.

Combining SSI verification with AA provides Ethereum a customizable authorization logic that can leverage national data repositories and business and regulatory standards before easily executing global on-chain operations. In this context, embedding a trust less verification logic can support standardizing user operations and streamlining sequencing activities, and so harmonizing operations across L2 transactions for unlocking cross chain atomic transaction and still ensuring efficient data processing compliant with data minimization.

The transaction generation, submission and execution flow can be represented with an interchained process of triangles of trust representing each moment as described here.

[

1600×900 42.6 KB

](https://ethresear.ch/uploads/default/original/3X/9/5/957433820f871fd453df50f9074cc21fd4bd9e1e.png)

The market pushing for an evolution of networks towards to modular blockchains is proposing concepts of shared mempools and shared sequencing that can embed trustless verification logic typical of SSI systems to establish a root of trust from user authentication until block validation still relying on Ethereum main network as indisputable security layer.

This combination minimizes data disclosure while maintaining efficient verification, making the system more scalable and privacy-preserving.

Key store contract for flexible and efficient key management system

To further enhance standardization and interoperability considering key management an important function to manage user authentication and transaction authorization. A keystore contract paired with zero-knowledge proofs (ZKPs) has the potential to significantly improve the verification logic and overall functionality of data assertions within the ERC-4337 framework.

The keystore contract can securely store and manage encryption keys used by bundlers to encrypt user operations within bundles after authentication. This can ensure that only authorized parties (with the appropriate decryption key) can access the sensitive data within the bundle.

[

image

1970×695 60.4 KB

](https://ethresear.ch/uploads/default/original/3X/4/2/42da34932f49f48f96077877f991029db0d43fb4.png)

The keystore contract can facilitate the pairing of the correct key with specific data assertions within a bundle. This allows different network participants (builders and sequencers) to efficiently verify the authenticity of the data using the corresponding ZKPs without needing the actual data itself. A minimal keystore rollup deployed on Layer 2 can handle a large volume of key management and verification processes off-chain, reducing the load on the main Ethereum network. The keystore functionality is key in order to ensure efficient authentication to users Identifiers and enabling cross chain user operations also leveraging shared sequencing. This allows for seamless data exchange and verification between different ecosystems, fostering a more interconnected environment supported of atomic cross chain transaction data process minimization and standardization. Different potential design for Keystore emerged in the market. Networks like Safe, Arbitrum, ZkSync, Scroll, Starknet have proposed different implementations.

L1 Keystore with L2 Sync

- This design keeps the keystore on Ethereum L1 and syncs with L2 networks to ensure quick access and state consistency across chains. It provides high security but may involve higher latency and costs.

When the keystore state is maintained on L1 and synchronized with L2 networks to ensure quick access and state consistency across chains. This approach maintains the security guarantees of L1 while aiming to minimize latency and reduce costs associated with high transaction fees on L1. This method provides a robust security framework due to the inherent security of L1, ensuring that key management and authentication logic are protected against potential threats. However, this design can involve higher transaction costs and potential latency issues due to the need to synchronize state changes across multiple layers. A reference implementation for this design can be seen in projects like Safe.

Dedicated L2 Keystore

- Deploying the keystore directly on an L2 network reduces costs and latency. This design uses zk-Rollups for secure key updates and storage. Deploying the keystore directly on an L2 network focuses on reducing operational costs and transaction latency. This design option utilizes zk-Rollups for efficient and secure key updates and storage, ensuring scalability without compromising security. By keeping the keystore operations within the L2 environment, transaction fees are significantly lower, and the response times for key management operations are improved.

Hybrid Model

- Combines L1 and L2 keystore advantages, using L1 for storage and L2 for efficient access and operations, leveraging cross-chain compatibility for seamless integration. The hybrid model combines the security benefits of L1 storage with the operational efficiency of L2 transactions. In this model, the keystore data is primarily stored on L1, ensuring high security, while L2 is used for accessing and updating keys. Merkle proofs are essential for maintaining consistency and verifying that updates on L2 are accurately reflected on L1. This model uses Merkle proofs to ensure that any changes made on L2 are consistent with the L1 state, providing a balanced approach that leverages the strengths of both L1 and L2. This design requires efficient generation and verification of Merkle proofs across both layers to ensure low latency and high security, making it a complex but highly effective solution for comprehensive key management.

Merkle proofs play a vital role in maintaining the integrity and consistency of keystore operations across different blockchain layers.

Here is a potential process workflow and designed architecture:

The architecture is composed by three main layers:

1. An application layer comprises wallets, PKIs apps, and SSI wallet such as Polygon ID, ZkSync ID, and other apps. These wallets and applications generate Verifiable Credentials (VCs) to prove ownership, such as being a DAO member. Users create operations as user intents via bundlers, enabling transactions and interactions on the blockchain. This layer facilitates the generation and management of credentials and user operations as “user intents” through apps, serving as the interface between users and the underlying blockchain infrastructure.

2. A network layer based on different L2s, which include a Keystore contract and Smart Contract Accounts. This layer is responsible for generating Zero-Knowledge Proofs (ZKPs) for bundlers and Merkle proofs for Sequencers. The Keystore contract manages encryption keys and user authentication, ensuring secure handling of credentials and operations. Smart Contract Accounts validate user operations, generating cryptographic proofs to ensure the integrity and validity of transactions before they are submitted to the blockchain for execution.

3. A sequencing layer which interconnect L2s with Ethereum main-net and manages the execution of batches of transactions anchoring Roll-up IDs to sequencing networks (projects like SUAVE, Polygon Agg Layer, Espresso Capuccino). Main functions here are batching, validation of transactions via the Keystore roll-up, and the Roll-up contract within Ethereum’s slots. This layer ensures efficient transaction processing by batching multiple operations into single transactions, reducing on-chain congestion and costs. Additionally, it enable cross-chain atomic transactions, enabling seamless interoperability across different blockchain networks, ensuring that transactions are securely validated and finalized.

[

1563×893 51.7 KB

](https://ethresear.ch/uploads/default/original/3X/6/7/676efe42126406535aece6ca6d086ff7e6396736.png)

1. User Authentication: Users authenticate using Verifiable Credentials (VCs) . They share Zero-Knowledge Proofs (ZKPs) of their credentials through a smart contract registry and sign user operations on a dedicated bundler.

2. Bundler Operations: Bundlers access a shared mempool ID, create bundles of verified user operations, establish a dedicated Merkle tree for each bundle post-authentication, and share the Merkle proofs and ZKPs with the Sequencer. They submit the transaction bundles to the Smart Contract Account (entry point).

3. Sequencer Verification: Sequencers verify Merkle proofs and ZKPs related to VCs. They validate bundles of transactions through the Smart Contract Account and submit them to the Block-builder for inclusion in a new block.

4. Keystore Rollup: The keystore contract and rollup manage key functions, including user authentication, transaction submission, validation, and key recovery.

5. Batching Transactions: Sequencer batch transactions into new block of Roll up IDs and submit them in batches via the Rollup contracts for validation on L1, verified by ZK validity proofs.

6. Rollup Validation: The Rollup contract validates ZK proofs of batches, and the block is finalized or reverted if validation fails.

Specifically in this scenario, proposals as ERC 5792 acts as a crucial bridge between user-facing applications and the infrastructure of the ERC-4337 framework. The proposal defines a common language for wallets, apps, and bundlers to communicate each other and it has the potential to significantly enhance the adoption of Smart Contract Accounts and position as the go-to standard for data assertions on the Ethereum blockchain.

Wallets and apps could signal user preferences (e.g., gas fees, preferred chain ID, or shared sequencer), required data assertions (VCs/EAs) for specific transactions.

Bundlers could leverage the received signals to automatically construct appropriate bundles for user transactions including user operations data, Merkle proofs for associated VCs/EAs and ZKP of user data claim. With this having an overall improvement for user flexibility, network decentralization and scalability.

The designed architecture also benefit from ERC-7715, further enhancing accounts functionalities on dedicated permission sessions for EOAs and SCAs aligniing with the EIP-7702 on Pectra. In fact standardized user authentication method will reinforce user protection about what permissions are being requested by dApps for starting sessions.

Once authenticated users were given permission to start sessions, dapps could leverage User operation builders through apps hosted on bundlers and following ERC-7679 compile different intents as standardized user operations.

ERC-7679 provides a common on-chain interface, the UserOperationBuilder, for dApps to interact with ERC-4337 Smart contract Accounts. This interface helps standardize how user operations are constructed, making it easier for dApps to support various account implementations without needing account-specific SDKs.

The proposed design also goes into the direction of modular networks that can be built over Ethereum network by the Ethereum community. In this modular scenario ERC 4337 and future developments benefits from possible future implementations as Verkle trees or consensus related proposals as ePBS.

Bundlers create bundles containing user operations, Merkle proofs, VCs/EAs (potentially with ZKPs), and additional data.

Sequencers verify Merkle proofs and potentially VCs/EAs within bundles, order bundles, perform additional verification if needed, submit pre-built blocks to validators.

Before accepting the request, the sequencer should:

- If block range was given, check that the block number is within the range.
- If timestamps range was given, check that the block's timestamp is within the range.
- For an address with a storage root hash, validate the current root is unmodified.
- For an address with a list of slots, it should verify that all these slots hold the exact value specified.

The sequencer should REJECT the request if any address is doesn't pass the above rules.

Builders construct valid blocks based on pre-verified and potentially pre-ordered bundles received from sequencers, considering block gas limits and network conditions. Ultimately, validators verify the validity of submitted blocks (including the validity of the included bundles), reach consensus on block inclusion in the blockchain.

CONCLUSION

Driving towards the adoption of blockchain services implies relying on modular networks that can balance security, scalability and decentralization to populate an ecosystem of applications as plug and play solutions for dedicated use cases, in this regard combining SSI verification logic into the network operations to users and machines to support accountability and compliant data processing process is key. Smart Contract Accounts can facilitate adoption by flexible UX and shared sequencing networks support cross chain atomic transaction reconciliation. The narrative of this research wants to emphasize the need to abstract complexities from accounts, and networks where decentralized identifiers and keystore roll up can facilitate the verification and execution of sequencing networks.

Takeaways

:

- SSI

: Integrates trustless identification, enhancing privacy and data minimization.

- AA

: Simplifies user experience and facilitates regulatory compliance and governance needs.

- Verifiable Credentials (VCs) and Ethereum Attestations (EAs)

: Used to empower bundlers, reducing the risk of on-chain reverts and penalties for block builders by providing cryptographic proofs.

- Merkle Proofs

: Efficiently verify specific data elements within larger sets, mitigating mempool pressure and lowering gas costs.

- Zero-Knowledge Proofs (ZKPs)

: Enhance privacy by proving the validity of claims without revealing underlying data.

- Shared Mempools

: Facilitate bundler operations and reduce computational burdens.

- Keystore rollups

Enhance atomic cross-chain transactions across sequencing networks by providing a scalable and interoperable key management solution that ensures seamless data exchange and verification across different L2.*

References

:

Notes on the Account Abstraction roadmap

RIP 7560

EIP 7562

Roadmap for Native Account Abstraction Introduction)

Unified ERC-4337 mempool

Blockchain Privacy and Regulatory Compliance: Towards a Practical Equilibrium

Keystore Design

Dedicated minimal rollup for keystores

Integration API for EIP-4337 bundler with an L2 validator/sequencer

Decentralized Future: ERC-4337 Shared Mempool Launches on Ethereum

Verifiable Credentials Data Model v2.0

Ethereum Attestation Service

Supercharging Account Abstraction with Attestations

Minimal KeyStore Rollup spec

ERC-1271: Standard Signature Validation Method for Contracts

EIP 7702

ERC-7715: Request Permissions from Wallets

ERC-7679: userOperationBuilder - a common onchain interface for dapps to interact with 4337 wallets

Towards the wallet endgame with Keystore

Keystore design by Haichen Shen