Project ID: 11

Project Title
Solstice: State Slicing For Storage Optimisation In Account-Based Blockchains

Client Name
Sushmita Ruj, Nhi Nguyen

Group Capacity
3 groups

Comment: This client might update or provide more details later or during first client meeting.

Project Background
Ethereum archive/storage nodes require ~21 TB of storage and grow by 1.2-1.8 TB annually. The current state data is inefficiently stored in a Merkle Patricia Trie (MPT), which requires the storage node to maintain all account information at every single state. As a result, this causes a high storage cost.

Our proposal – SOLSTICE optimises the data storage for account-based blockchain (e.g., such as Ethereum) using vector commitment (**see below**)

** The Innovation ** Reduces the storage cost by 80%

** Project Goals **
- Implement SOLSTICE with vector commitment (VC) schemes
- Evaluate SOLSTICE with different parameters setting
- Implementation that can be served as an alternative design for archive nodes

Project Scope
SOLSTICE is an alternative design for archive node (i.e., which stores blockchain historical data). We aim to:

- Implement an extended version of SOLSTICE that supports different VC schemes
- Evaluate SOLSTICE against existing archive node implementations
- Testing with Ethereum transaction data (e.g., less than 1000 blocks)

** Timeline **
- Read about the VC scheme, Ethereum data structures (2 weeks)
- Implement VCs /or work on existing VC scheme implementations (2 weeks)

- Fetch Ethereum data (1 week)
- Implement and evaluate SOLSTICE (4 weeks)
- Testing on Ethereum data (1 week)

Project Requirements

- Generic SOLSTICE Implementation: Rust/Golang codebase supporting different VCs (e.g., Reckle Tree and Hyperproof)
- Benchmarks: Automated testing

- Storage: Measure total disk space used by SOLSTICE vs. traditional Ethereum archive nodes
- Performance: Proof generation and verification time, proof size
- Comparative: Our scheme with different VCs vs. existing implementations

Note: We have implemented a basic version of SOLSTICE with Hyperproof in Golang.

Required Skills

** Essential **
- Strong knowledge on Data Structures and Algorithms
- Excellent programming skills in Rust and Golang
- Desirable: Knowledge of Cryptography and Blockchain (Ethereum)

Expected Outcomes

- Implement SOLSTICE in a modular design that users can easily "plug in" different VC schemes (if needed).
- Support at least two VC schemes
- Detailed evaluation
- Clear documentation and clean code

Disciplines

Computer Science and Algorithms; Blockchain and Cryptography ;Security/Cyber Security;

Other Resources

Link to papers and their implementations:

** Ethereum **
- https://ethereum.org/en/developers/docs/data-structures-and-encoding/

** Hyperproof **
- https://eprint.iacr.org/2021/599
- https://github.com/hyperproofs/hyperproofs-go

** Reckle Tree **
- https://eprint.iacr.org/2024/493
- https://www.youtube.com/watch?v=lcWQHYox0qc
- https://github.com/Lagrange-Labs/reckle-trees