Zero-knowledge authenticated data structures

implementation on o1js

RSSCNo1

Progress overview

- Pre-investigation of current o1js library
  - from Mina docs and source code
- Design of stateful-ZKADS prototype
  - succinct witness per operation type & authentication by ZkProgram
- Research work focusing on existing ZKADS protocols and their compatibility with the o1js library
- Testing and benchmarks of the Merkle-tree module
  - Still failed ☹

Stateful-ZKADS prototype

- Database, built on some ADS schemes, abstracted as **states** with specific transition functions
  - for static structure, transition functions to maintain recorded-on-node value and labels
  - for dynamic structure, transition functions to maintain recorded-on-node values and labels, as well as the adjustment of links between nodes
- Every operation results in a witness ready for ZK verification
  - operation parameters and witness as private inputs of the circuit
  - in case of recursive proofs, proof of previous step also involved
  - must be succinct enough to suit the ZK-computation capacity

ZKADS protocols: sets and much more

- Most solutions are still limited within (static or dynamic) sets:
  - **ZK-Set**: a fixed set, membership or non-membership of a value
  - **ZK-Accumulator**: enabling element insertion & deletion on ZK-Sets
  - **ZK-Indexed-Set**: each element of a ZK-Set with an index
- Extension: from sets of isolated elements to integrity
  - supporting operations referring to a bunch of elements
- FHE-friendly hashes in demand
  - as operations may perform calculation on the stored data

Off-chain modules

- Stateful ADS interface as operator
  - perform access and update operations on the database
  - generate witness for ZK verification
  - export API for each type of operation
- ZkProgram as verifier
  - verify the result of an operation by a circuit
  - input: operation parameters, previous state proof (recursive), witness
  - output: the new state proof

On-chain stored values

- The global checksum
  - ensure the genuineness of each off-chain storage
  - e.g. the root hash of the world-state Merkle-tree
- Other values maintained by the entire network
  - e.g. the synchronized timestamp
- Used in ZkProgram verifier as **public inputs**
- Bottleneck: compacted into 8 Field (int256) numbers
  - external recorded and hashed on-chain?

Tests and benchmarks

- Prepare the formalized stateful-Merkle modules
  - `MerkleWitness` seems not provable (rewritten in a provable manner)
  - `bigint` is not provable, and no usage referred to `BigIntField`
- Testing value update and access in an o1js Merkle-tree
  - failed, unknown internal issue of the circuit
- Benchmarking the circuit
  - not managed to retrieve the size of relevant circuits and proofs
  - stuck on the provability issues
- Technical help required to pass the tests...

Thank you