Proof of Achievement M3

Protostar prover algorithm

The Protostar algorithm implementation is split across several modules, which can be found here

https://github.com/zkFold/zkfold-base/tree/vks4git/protostar/src/ZkFold/Base/Protocol/ARK/Protostar

Protostar folding allows construction of "recursive" arithmetic circuits. These circuits correspond to programs containing loops and recursive function calls. On top of that, it also enables creation of data types with variable byte length which are very common in the Cardano ledger specification.

This feature makes our zkFold Symbolic framework much more robust and enables efficient implementation of zero knowledge smart contracts on Cardano.

zkFold Symbolic transaction balancing algorithm

We have also implemented off-chain (transaction balancing) code for zkFold Symbolic smart contracts. One can find the relevant console scripts here:

https://github.com/zkFold/zkfold-cardano/tree/main/e2e-test/plonk

https://github.com/zkFold/zkfold-cardano/tree/main/e2e-test/symbolic

Besides that, the off-chain Haskell code related to the task can be found here:

https://github.com/zkFold/zkfold-cardano/tree/main/backends

Currently, we have two types of zkFold Symbolic smart contracts:

- Plonk smart contracts mint a token if and only if a proof verification succeeds on-chain. The input to the verification algorithm is the token name.
- Symbolic smart contracts succeed if and only if a transaction satisfies a particular condition encoded as an arithmetic circuit.