VeriSmart

is a tools that permits migrating Smart Contracts between EVM-compatible blockcains, maintaing it's execution logic and state. No trust in the migrating entity is required, as the target contract's validity is verifiable through merkle proofs.

The details of the chosen approach can be found in a research paper that is going to be published at ICBC2019: https://arxiv.org/abs/1902.03868

A prototypical implementation can be found here (a bit messy still, but works as a start): https://github.com/informartin/VeriSmart

Motivation:

During a smart contract' life cycle, external requirements may change or the host blockchain may become insecure due to e.g. lacking incentives, bugs etc. Changing application scenarios may also require shifting from a public chain to a permissioned one or vice versa.

Contrext:

the contract code needs to be executable on both platforms and support for as many platforms as possible is desirable to enable full flexibility in choosing the target chain. Recently, many blockchain implementations have adopted the EVM as a common execution environment (e.g. Hyperledger Sawtooth, Hyperleder Fabric, Hyperledger Burrow, Quorum, Qtum and Counterparty).

Approach:

To enable smart contract portability between EVM-compatible blockchains without trust requirements in the executing enitity, I have come up with a mechanism that re-executes all past transactions to reconstruct the currect contract state. While doing so, a key/value map is built that represents the current state at the time of the transaction's execution. After re-executing all past transactions chronologically, the final state is obtained and set in the target contract's constructor. Using the resulting merkle tree, the state's validity becomes verfiable (i.e. no alterations have been made to either contract code or state). Depending on the contract's size, multiple approaches exist to achieve this. For instance, the contract state may be too large to be deployed in the constructor of the contract. In this case the contract is split into deployment-, logic-, and state-contract. The state contracts holds all variables, but delegates function calls to the logic contract. The deployment contract is authorized to set variables during the deployment phase and is destroyed thereafter to prevent unintendet alterations. Also, many smart contracts refer to other contracts or inherit from them, in these cases dependencies have to be deployed recursively while inserting the new addresses in the bytecode/state. All operations are conducted in bytecode so that the original high level code is not required.

I would appreciate any thoughts and feedback!