My problem is rather simple. I want to create a smart contract which will hold a merkle root committing it's state with internal book-keeping. Clients who interact with this contract will need to send a merkle branch on which their action is based.

Lets say that the internal book keeping consists of 1024 data-points of uint256

type.

Most obvious approach that comes to me is to use as hashing function H_{1}(a,b) = keccak256(a, b)

, which returns unit256

. A client who wants to prove to smart contract a data point will need to provide 640 bytes, calculated as:

2 \cdot 10 \; \textrm{levels} \cdot 32 \; \textrm{bytes} = 640 \; \textrm{bytes}

640 bytes is not terrible, but I want to make the system as gas-efficient as possible, so I'm looking to possibly use smaller hashes.

Somewhat attractive approach would be to use as hashing function H_{2}(a,b) = bytes4(keccak256(a, b))

.

For H_{2}

the size of the proof for merkle tree with 1024 leafs would be:

2 \cdot 32\;\textrm{bytes} + 2 \cdot 9 \; \textrm{levels} \cdot 4 \; \textrm{bytes} = 136 \; \textrm{bytes}

Which sounds much better.

So I have few questions regarding the scheme above, perhaps someone can point me to some research on the subject.

1. Is H_{2}

secure enough, can an attacker efficiently falsify a merkle branch using brute force ?

1. If H_{2}

is not secure, does anyone know what is practical minimal length of hash that can be considered secure for this problem ?

1. Perhaps someone knows even better hashing function for this problem? By "better" I mean more secure and gas-efficient.