This was inspired by: https://github.com/ethereum/stateless-ethereum-specs/issues/30#issuecomment-636106478

A quick look into account and storage values for compressability in the context of block witnesses.

It is worth noting that everything below is leaf

data, which represents only a small percentage of the overall witness size, and thus, while we can realize some significant gains for certain fields, overall this will have a minimal effect on the overall size of the witness which suggests we take the easy wins and ignore anything that introduces too much complexity (such as compression of storage values).

Storage trie key for an Account

The key in the account storage trie where the account data is located.

- raw: keccak(address)

-> 32 bytes

- compressed: address

-> 20 bytes

Account.balance

The balance

component of the RLP account:

- raw: 32-byte big endian encoded integer: 32 bytes
- compressed: LEB128 for different thresholds:
- 0 - just a few wei: 1 byte
- 1 ether: 9 bytes
- 10 million ether (bitfinex has ~4mil ether): 12 bytes
- 0 - just a few wei: 1 byte
- 1 ether: 9 bytes
- 10 million ether (bitfinex has ~4mil ether): 12 bytes

There should not be any major pathological edge cases since users cannot set arbitrary balances and even extremely high balances save space over the full 32 byte version.

Account.nonce

The nonce

component of the RLP account

- raw: 32-byte big endian encoded integer: 32 bytes
- compressed leb 128:
- 1 byte for most accounts
- 4 bytes or less for accounts with nonce < 10 million
- 1 byte for most accounts
- 4 bytes or less for accounts with nonce < 10 million

There should not be any major pathological edge cases since users cannot set arbitrary values for the nonce.

Account.code_hash

The code_hash

for the account

- raw: 32-byte hash
- compressed:
- ~0 bytes for EOA accounts if we use empty string (plus maybe 1-2 bytes for serialization overhead)
- not compressable for contracts
- ~0 bytes for EOA accounts if we use empty string (plus maybe 1-2 bytes for serialization overhead)
- not compressable for contracts

Account.state_root

The state_root

for the account

- raw: 32-byte hash
- compressed:
- ~0 bytes for EOA accounts if we use empty string (plus maybe 1-2 bytes for serialization overhead)
- not compressable for contracts
- ~0 bytes for EOA accounts if we use empty string (plus maybe 1-2 bytes for serialization overhead)
- not compressable for contracts

Contract storage keys and values

For keys, I suspect there is little that can be done to compress them. In theory some contracts primarily use keys with very small integer equivalents, but the majority of keys likely use the full 32-byte keyspace (can anyone validate this assumption?)

For values, there may

be room for improvement via compression. For example, ERC-20 token contracts would have values that primarily fall in the numeric range that would allow significant compression gains via something like LEB128. However, the pathological case where values take up the full 32 bytes space would result in 37 bytes per value (15% increase in size). Quantitative research would be needed to determine whether this would actually be worth doing.