Alternative UTXO set proposals

Alternative UTXO proposals

- Alan Reiner’s ‘trust-free lite nodes’ (2012)
- Peter Todd’s TXO MMR commitments (2016)
- Bram Cohen's TXO bitfield (2017)
- Pieter Wuille’s Rolling UTXO set hashes (2017)

trust-free lite nodes (Reiner)

- Proposed by Alan Reiner in a bitcointalk post
- There is a second chain that commits to the UTXO set, which is merge-mined with the main chain.
- The commitment is some kind of tree, perhaps a binary search tree, or a patricia tree, or a de-la-briandais tree, or a red-black tree.
- Lots of discussion about what this structure should be. If the UTXO is to be committed to, it should be fast to update (or committed to in a later block)
- Lite client stores the root of this structure and asks for proofs-of-inclusion / proofs-of-exclusion for their outputs.

trust-free lite nodes (cont)

- You can just download a utxo set from somewhere, and check using the latest block that it's valid (or, say, 1000 blocks back, depending on how certain you want to be).
- This discussion evolved towards the root being a commitment instead of an alt-chain
- Pieter Wuille had implemented ultraprune at roughly the same time. He suggested that Ultraprune could be thought as a step towards these ideas

TXO MMR commitments (Todd)

- A new hash root is committed to in the block. The structure is a Merkle Mountain Range:
  - deterministic
  - indexable
  - insertion ordered
- New items can be cheaply appended to the tree
- Once items are added, they are never removed, just updated in place.
- The state of a specific item in the MMR, as well as the validity of changes to items in the MMR, can be proven with log2(n) sized proofs consisting of a merkle path to the tip of the tree.

TXO MMR commitments (Todd)

- At an extreme, with TXO commitments we could even have no UTXO set at all.
- A more realistic implementation is to have a UTXO cache for recent transactions.
- The spender provides the proof that the TXO exists and is unspent.
- Proofs can be generated and added to transactions after the fact
- The commitment is delayed - ie in block *i* the miner commits to the TXO set in block *i-n* where *n* is some system constant.

TXO MMR commitments (Todd)

- A later post suggested that the TXO commitment doesn’t need to be committed at all
- A full node that doesn't have enough local storage to maintain the full UTXO set can instead keep track of a TXO commitment, and prune older UTXO's from it
- In the event those UTXO's are spent, transactions and blocks spending them can provide proofs to temporarily fill-in the node's local TXO set database

TXO bitfield (Cohen)

- Wallets maintain a proof of position for each output they want to spend (a simple merkle proof off some block, but with an output number rather than just a tx number), which is immutable.
- Nodes maintain a TXO bitset, which indicates what outputs are spent: even implemented as a naive bit array, this is 1/256 the size of the full TXO set
- Over time, the bitfield becomes sparse, and can be compressed quite compactly.
- This is not a block consensus change, it’s merely a peer-to-peer upgrade.

Rolling UTXO set hash (Wuille)

- Concatenating all the UTXOs in a canonical order and then hashing is simple, but expensive to update.
- Efficiency requires that we be able to hash the data in any arbitrary order, and remove as well as add elements.
- Proposal initially not for committing data. Use cases are:
  - Replacement for Bitcoin Core's gettxoutsetinfo RPC's hash computation.
  - Assisting in implementation of schemes like assumeutxo
  - Database consistency checking

Rolling UTXO set hash proposals

- A rolling, incremental hash of the set of objects.
- ‘Hashing onto a curve point’ on the libsecp256k1 curve. Hash each item in the set onto the curve, and then sum them under elliptic curve addition
- Lthash - homomorphic hashing

Rolling, incremental hash

- Bellare-Micciancio paper “A New Paradigm for Collision-free Hashing: Incrementality at Reduced Cost” suggests ways of incremental hashing:
    - XHASH - hash individual objects and XOR - is trivially insecure under Wagner’s attack
    - AdHASH - hash individual objects and add modulo some large prime - is also insecure under Wagner’s attack.
    - MuHASH - hash individual objects and multiply modulo some large prime - is secure under the DL assumption.
- If we use MuHash, we remove an item by finding the inverse of its hash under multiplication modulo the prime.

Hashing onto a curve

- Elliptic Curve Multiset Hash is efficient, but uses a strange binary elliptic curve
- One other approach is just reading potential X coordinates from a PRNG until one is found that has a corresponding Y coordinate according to the curve equation. On average, 2 iterations are needed.
- Then add the points under EC point addition. To remove a point from the set, add its inverse (same point with Y co-ordinate inverted)

LtHash

- Developed by facebook security team
- All data elements are hashed to a 2KB digest
- Two digests can be ‘added’ by breaking up each output into 16-bit chunks and performing component-wise vector addition modulo 2^{16}.
- Properties:
    - Set homomorphic
    - Collision resistant

Further reading

Further reading

- **UHS** - Full-node security without maintaining a full UTXO set
- **utreexo** - A dynamic accumulator for Bitcoin state
- **Accumulators** - A scalable drop-in for Merkle Trees
- **flyclient** - Super-Light Clients for Cryptocurrencies

Questions?
Comments?

chaincode