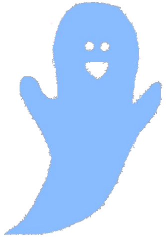
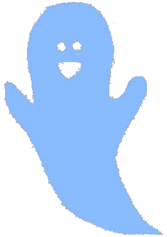
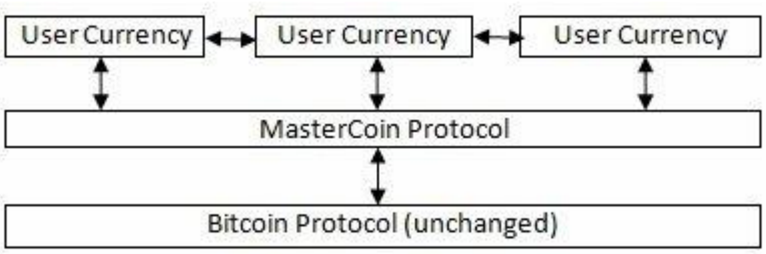
Scalable blockchains as data layers





Prehistory: meta-protocols

- Mastercoin: “meta-protocol” on top of the Bitcoin blockchain
- Uses the Bitcoin blockchain as a data store, NOT as a state execution engine
- Anyone can compute the Mastercoin state as follows:
  - Let t[1] … t[n] be an ordered list of all transactions in the Bitcoin blockchain
  - Let G be the genesis state, and STF be the Mastercoin state transition function
  - S[1] = STF(G, t[1])
  - S[2] = STF(S[1], t[2])….
  - S = S[n] = STF(S[n-1], t[n])

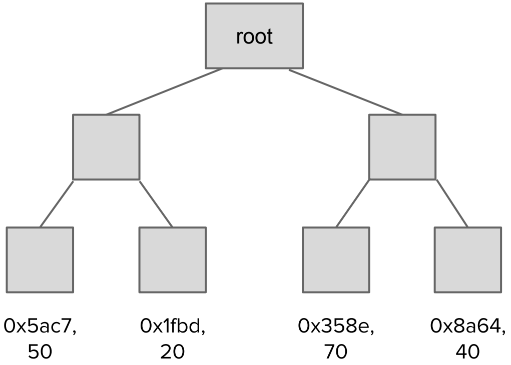


Mastercoin

Problems

- Not light client friendly
- MSC activity cannot influence any state outside of the MSC meta-protocol (eg. cannot handle bitcoins)

ZK Rollup



- Contract stores state as Merkle root of `{account ID => (pubkey, balance)}`
- Users send txs of ~13 bytes each plus signature

From	To	Value	Fee	Nonce	Signature
3 bytes	3 bytes	4 bytes	1 byte	2 bytes	32-96 bytes

- Relayer gathers set of transactions `t[1]...t[n]`, creates ZK SNARK:
    - `STF(PRE_STATE, t[1] .... t[n]) = POST_STATE`
    - Each `t[i]` has valid signature
    - `root(PRE_STATE) = r1`
    - `root(POST_STATE) = r2`
- Relayer publishes `r1, r2, t[1]...t[n]` without signatures and SNARK. Contract verifies SNARK and that `root == r1`, and sets `root = r2`

ZK Rollup

- Cost of a rolled-up transaction: ~500k gas for a STARK, plus ~884 gas (68 * 18) per transaction
- Note: this scheme can process instant deposits/withdrawals and allows anyone to become a relayer, because unlike Plasma it is NOT susceptible to data unavailability attacks
    - Publishing data required to allow anyone to reconstruct the current balance tree
- Takeaway from Stanford eth1.x workshop: a gas cost reduction in the cost of data (now 68/byte) is likely!

Taking rollup further

- Support more complex state transition functions
- Multiple tokens
- Privacy-preserving computation
- All using the “SNARK + publish tree deltas” paradigm

ZK ZK Rollup

- Users publish txs with SNARKs saying “I have a valid spend certificate for some coin hash in the state. Here is a new coin hash”
  - Eg. user secret: `s`, coin hash: `h(s + 1)`, spend certificate: `h(s + 2)`
- Transaction does not reveal which coin it is spending. However, each spend certificate can only be used once...
- Relayer only publishes a receipt

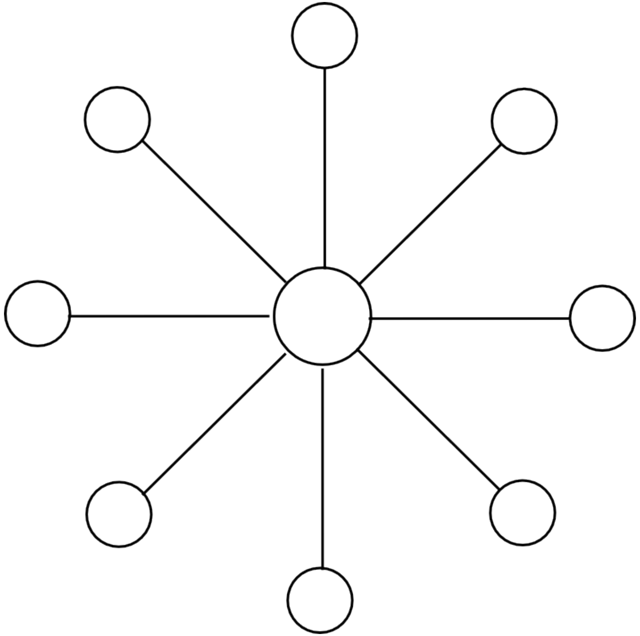
Spend certs	New coin values	Fee	New hash
h1: 32 bytes	o1.value: 4 bytes	1 byte	32 bytes
h2: 32 bytes	o2.value: 4 bytes		

Only one hash required for N outputs because it’s the Merkle root of all the output coin hashes; the spend would need to provide the Merkle branch to the relayer

- Receipt sufficient to allow anyone to construct updated state
- Gas cost: 105 * 68 = 7140 per tx

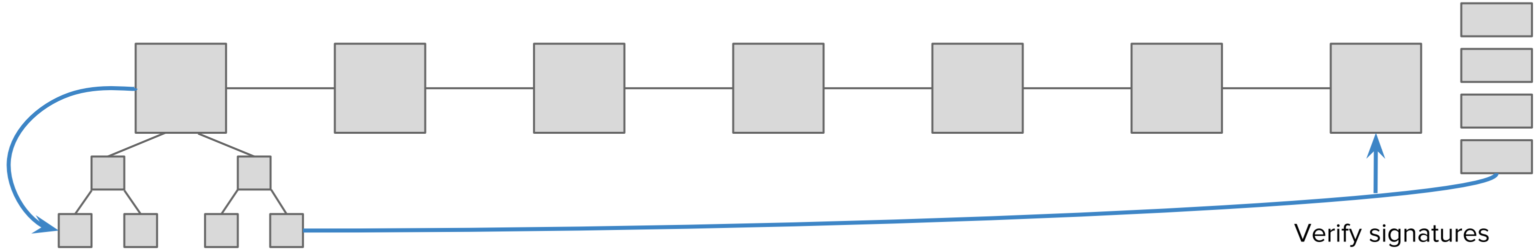
Beacon chain phase 1

- Shard chains as data-only chains
- ~2.8 MB/sec of data availability
- Let’s throw things on there so our layer 2’s don’t have data availability issues!



Eth2 in eth1 light client

- Light client only requires ~80kB of Merkle branches per 9 days, plus ~500 bytes per header
- Main challenge: need BLS-12-381 precompile in eth1



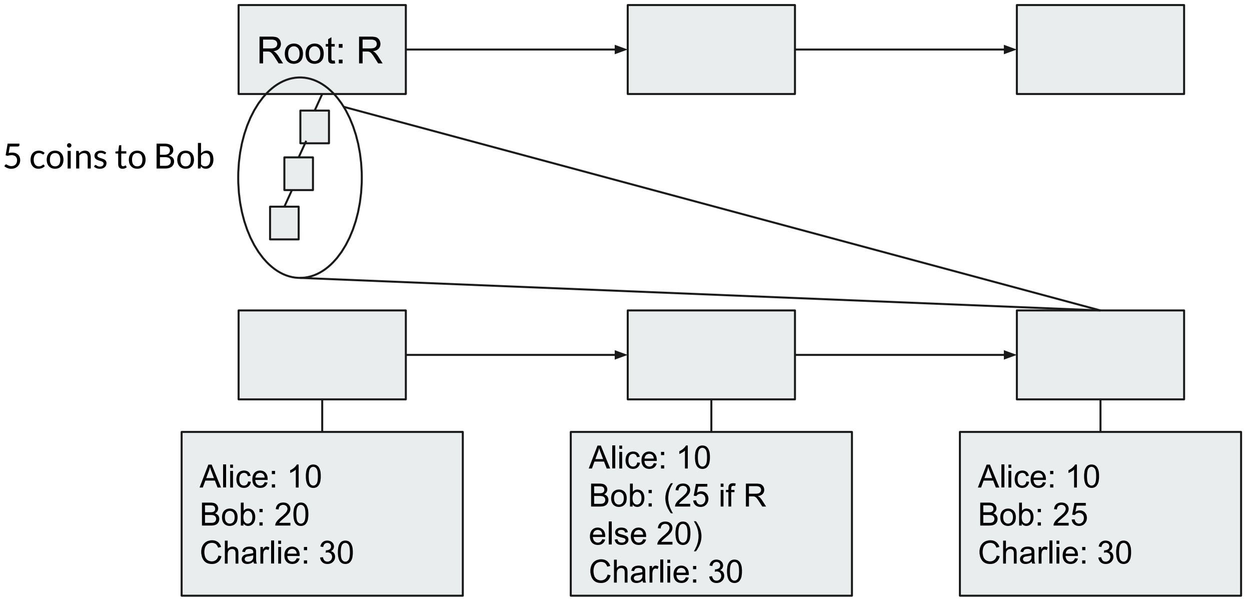
Authenticate validator set from state root in header

Verify signatures using older validator set to authenticate newer header

Other things you can do with scalable data availability engines

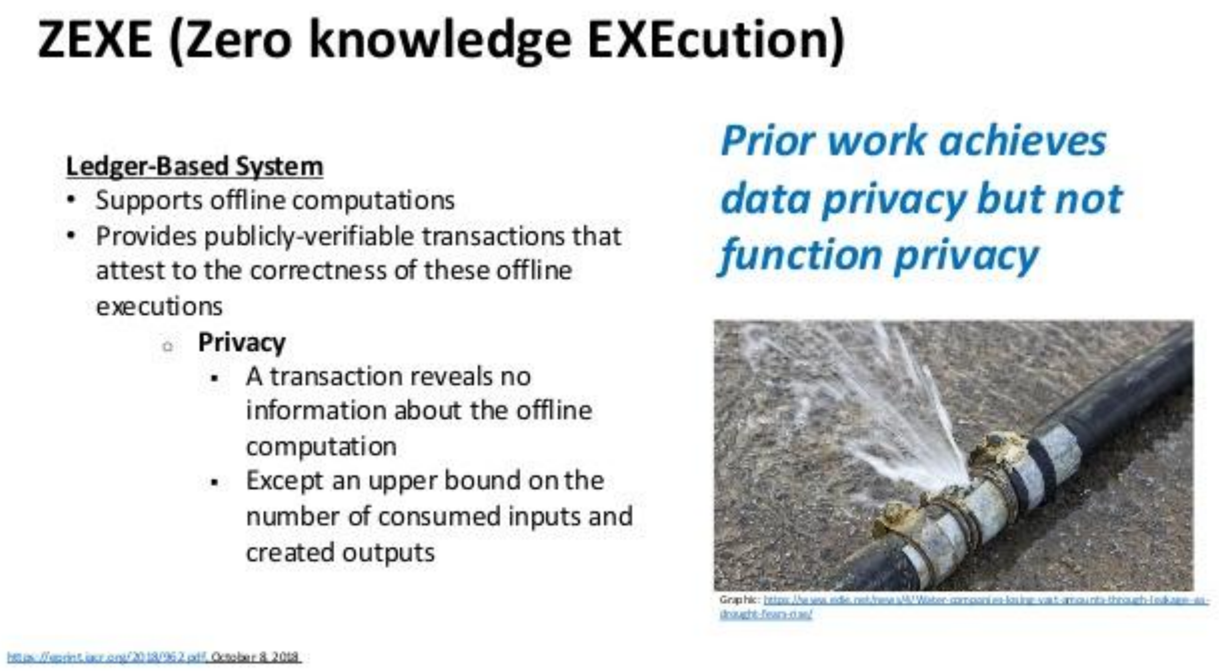
- Plasma chains with much more frequent commitments
- Dapps storing messages on-chain
- Blockchain protocols with independent (“sovereign”) state transition functions piggybacking on ethereum for data availability

Speeding up cross-shard transactions



- Store conditional objects in the state of each shard
- Resolve them as we later learn about the actual state roots and branches
- See: https://ethresear.ch/t/a-layer-2-computing-model-using-optimistic-state-roots/4481

General-purpose privacy



ZEXE (Zero knowledge EXEcution)

__Ledger-Based System__

- Supports offline computations
- Provides publicly-verifiable transactions that attest to the correctness of these offline executions
  - **Privacy**
    - A transaction reveals no information about the offline computation
    - Except an upper bound on the number of consumed inputs and created outputs

Prior work achieves data privacy but not function privacy

Graphic: https://www.edie.net/news/4/Water-companies-losing-vast-amounts-through-leakage-as-drought-fears-rise/

https://eprint.iacr.org/2018/962.pdf, October 8 2018

Benefits of the “layer 2 computation” paradigm

- Layer 1 does NOT need to over-complexify to optimize properties
- Layer 2’s can upgrade over time, less need for active governance of layer 1
- No need to do “ethereum 3.0 super-quadratic sharding”, can just keep using the beacon chain and increase shard count over time as tech improves
    - “Beacon chain as end of history” thesis