Review of Bitcoin Scaling Proposals

Bryan Bishop <kanzure@gmail.com>

0E4C A12B E16B E691 56F5 40C9 984F 10CC 7716 9FD2

2015-09-12

LedgerX

My approach

- An attempt at thorough review
- All bitcointalk.org technical (6.0) forum threads
  - https://bitcointalk.org/index.php?board=6.0
- All bitcoin-dev mailing list threads
  - http://lists.linuxfoundation.org/pipermail/bitcoin-dev/
- Some chunks of #bitcoin-wizards and #bitcoin-dev IRC logs, although not everything.
  - http://gnusha.org/bitcoin-wizards/
- Also sought input from ~20 suspects

Authorship statement

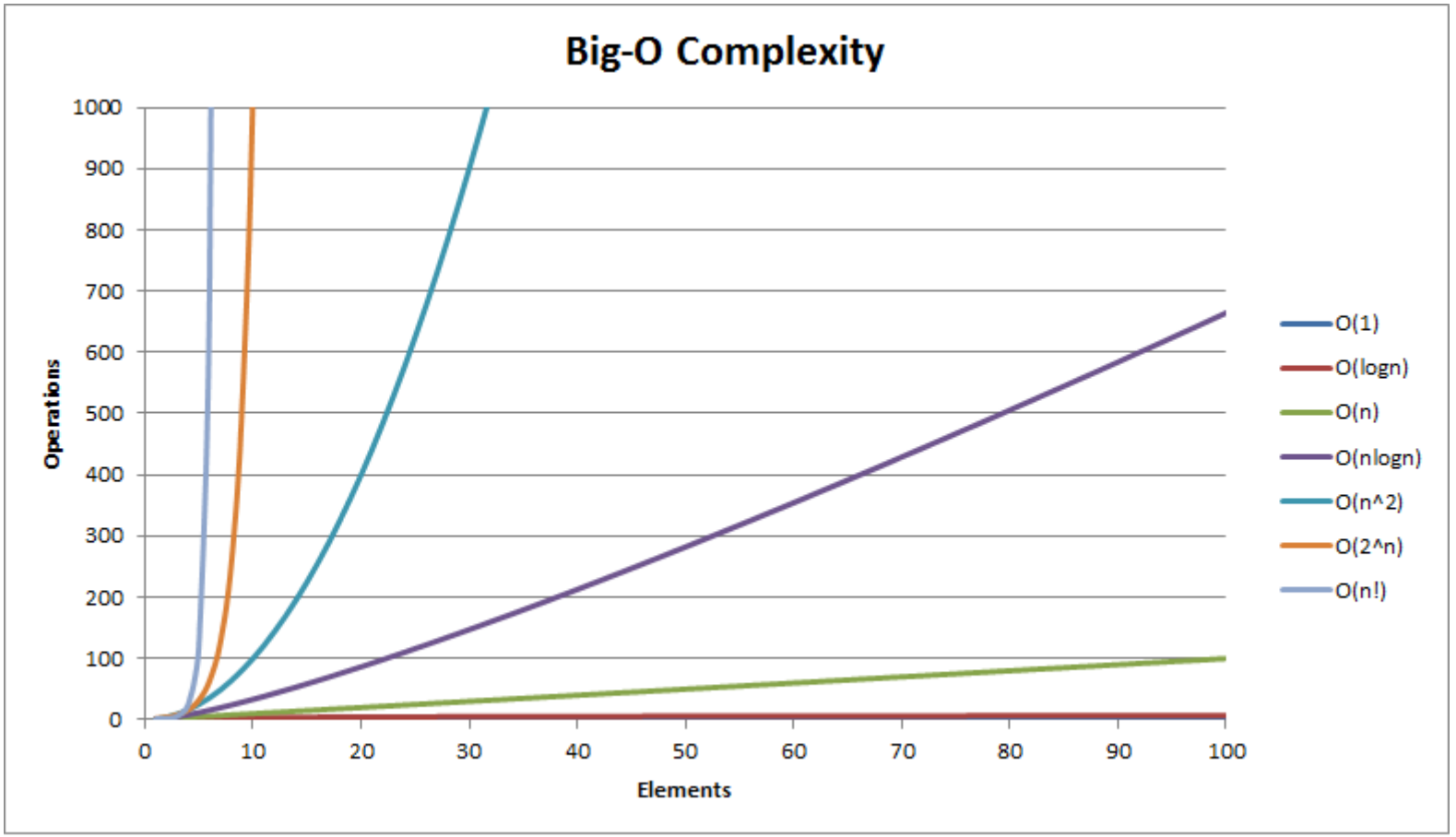
- This is all about work from the larger community, it's not my work
- Authors of almost all of these designs are here today at the scalingbitcoin.org workshop
- Most slides have links that show authors and contributors
- I highly encourage you to flag down the authors of the ideas you like, and coordinate with them

General observations

- Difficult to listen to all ideas
  - Easy to miss almost everything
  - Easy to lose good ideas
  - Signal-to-noise ratio
  - Use descriptive, unique names; context matters.
- Much of early focus was about slow initial sync of early blockchain
  - "80 MB blockchain takes too long to download"
  - Slow verification was partly responsible
- Scarce resources:
  - Software development effort
  - Review effort

Scaling

Name	Running time	Examples
constant	O(1)	even/odd
logarithmic	O(log n)	binary search
linear	O(n)	find min/max in unsorted integer array
log linear	O(n log n)	merge sort
quadratic	$O(n^2)$	insertion sort, bubble sort
exponential	$O(2^n)$	generalized chess


Big-O Complexity
Operations
1000
900
800
700
600
500
400
300
200
100
0
0
10
20
30
40
50
60
70
80
90
100
Elements
O(1)
O(logn)
O(n)
O(nlogn)
O(n^2)
O(2^n)
O(n!)

What's the theoretical max scale?

- ~10^{80} hydrogen atoms in observable universe
- Computronium (grey goo) scenario
  - Convert all matter in universe into transaction processing dark matter
  - Essentially same plot as every astronomical disaster scifi story
- Minimum energy necessary to flip 1 bit
- Physics of Information Processing Superobjects (Jupiter brains, etc.)
- Upload everyone's brain into cloud, analytically simulate all economic activity, problem solved. (ask Ralph Merkle!)
- How many transactions per second? Minimum/maximum?

Immediate large scale through indiscriminate centralization

- Millions of transactions/sec easily achieved
- 3.5 billion transaction verifications/sec per cubic foot of custom ASICs, 1 rack could handle 19 transactions/sec-person
  - 234 billion/sec/rack if using sha256 ASICs for lamport signature verification
- Single supernode, no network
- Replace blockchain with PostgreSQL database
- Digitally signed audit logs, like certificate transparency, no mining.
- Registered, verified users (not P2PKH)
- Offer chargebacks, reversible transactions, etc.
- BTC converted into ISOs, largest startup ever
- Much easier to comprehend, way better than modern banking :-(
  - it's awful and yet still better!

Now back to reality...

What's the "correct" scale?

- How many transactions does a civilization need?
- High transactions/sec may be unhealthy
- Supermassive Kardashev-2 civilization?
- OK for some humans to be uninterested in using bitcoin. Does all (non)economic activity need txns?
- Many ways to accidentally drive system off cliff.. permanently?
- Systemantics, S. Salthe, technium

Scaling what?

- Trustlessness, financial safety, mining, privacy, fungibility, … Ultimately these might be political issues.
- Transaction verification bandwidth/capacity
- Transaction reliability, network availability
- Node count, node usage
- Difficulty with traditional measurements in bitcoinland
  - Conflate transactions / users, low amount / unsolicited commercial advertisements
  - Can't measure node count, "identity", time, intentional mutants, user count, node size, mining costs, …

Tradeoffs?

- Speed/size
- Cost/benefit
- Efficiency/security
- Time/memory
- Privacy/storage
- Size/false positives (bloom filters)
- Bandwidth/variance (mining)
- Honestly-faulty/malicious
- Decentralization/scale? Trustless/comprehensible?

Byzantine security and scaling

- Best case / average case / worst case
- Best case O(n) not helpful if attacker can force a worst case O(n^2)
- Greater focus on improving worst-case performance

Some bottlenecks

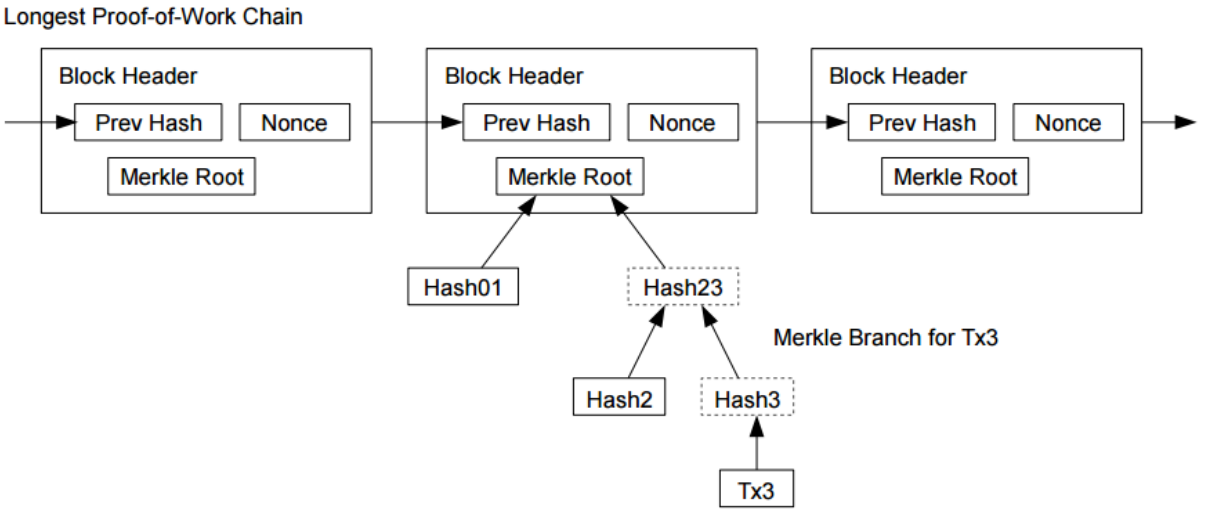
- Transaction verification
- Bandwidth
- Node costliness, node count
- Mempool size
- p2p flood/gossip network
- User onboarding, education, training
- Recovery from consensus hard-forks?
- Code review

Some data structures

- Distributed hash table (DHT)
- Merkle tree (hash tree)
- Merkle sum tree
- Merkle mountain range (insertion ordered binary tree)
- Merklized abstract syntax tree
- Bloom filters

Simplified Payment Verification (SPV)

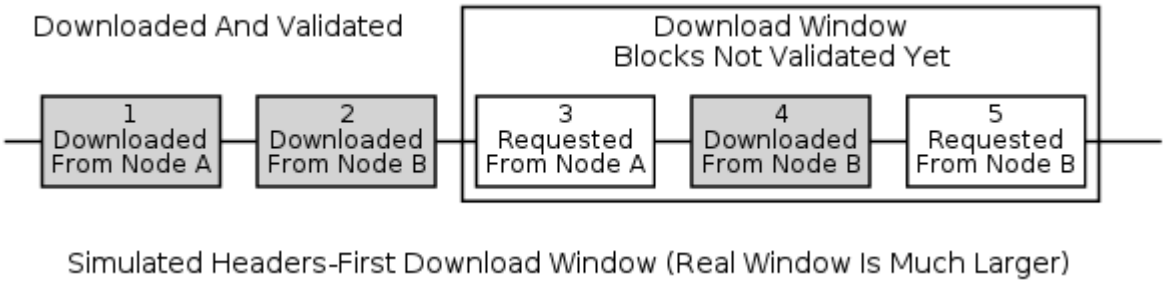
- Lightweight clients and not fully-validating
- High PoW difficulty as proxy for proof-of-validity
- Download headers-only; scales linearly with time since blockchain genesis.
- Use merkle trees for proof-of-inclusion of transactions; or merkle tree of unused output tree (UOT)



https://bitcoin.org/bitcoin.pdf

Headers-first

- An optimization for full nodes (fully-verifying)
- Previous solution was blocks-first
- Synchronize blockheaders first, partially verify, next download each block
- Vastly improved blockchain sync
- Merged into Bitcoin Core as of v0.10.0



https://bitcoin.org/en/developer-guide#headers-first

Pruning

- Local
  - SPV lightweight clients (as mentioned)
  - Nodes store rolling window of blocks (~1 GB)
  - Ultraprune - stores UTXO index, not TX index
  - UTXO pruning
- Global
  - physical blockchain pruning, use snapshots, etc.
  - OP_RETURN pruning- controversial, would eliminate certain OP_RETURN usecases, reclaim/save space
  - UTXO pruning- invalidate old UTXOs
    - burn old BTC
    - move old BTC to miner subsidy or other purposes

Bloom filters

- Zero false negatives in exchange for false positives
- Precision/bandwidth tradeoff for SPV nodes
- SPV node constructs bloom filter, give filter to larger p2p node for checking before transferring potentially irrelevant transactions (BIP37)

https://github.com/bitcoin/bips/blob/master/bip-0037.mediawiki

"Just use DHTs"

UTXO commitments

- Provide commitment in block header
- Avoids traditional UTXO sync ("validate entire blockchain")
- Transactions provide merkle path proofs for verification
- Soft-fork can require UTXO commitments to be valid, but the commitment itself optional
- There may be incentive problems with miners not validating commitments..

http://diyhpl.us/~bryan/papers2/bitcoin/wizards/2013/10/13-10-17.log

Other TX commitment ideas

- UTXO commitments (unspent-only)
- STXO commitments (spent-only)
  - insertion-STXO proofs might be more bandwidth-friendly
- TXO commitments (all)
  - can be insertion-ordered
  - doesn't support query-by-hash
- Various UTXO pruning proposals
- MMR TXO commitments- throw away most blockchain data, wallets provide utxos and proofs, etc.

Amnesic Fixed-Size UTXO Set Commitments

- Blocks contain commitment to constant/fixed size UTXO set, probably in block headers
- Prune old UTXOs
- Bandwidth-consuming proof for the occasional spend of old coins
- Merkle mountain range TXO proposals
- Fixed-size storage cost full nodes

Invertible Bloom Lookup Tables (IBLT)

- Mempool transaction set reconciliation
- New block announcements include less data except probably-unique transactions
- Faster block relaying across network
- Propagation is O(1) for transactions already seen by majority of network, and O(1) for blocks with same
- Requires cooperation, large miners have incentives not to cooperate

https://gist.github.com/gavinandresen/e20c3b5a1d4b97f79ac2
https://en.bitcoin.it/wiki/User:Gmaxwell/block_network_coding

Relay Network

- High-speed block-relay system for miners
- Strategically-placed nodes around the world
- 100-300 ms propagation
- Does not use p2p bitcoin network
- Does not replace p2p bitcoin network
- Only partial validation
- Actively used by miners and others

http://bitcoinrelaynetwork.org/
https://github.com/TheBlueMatt/RelayNode

MAX_BLOCK_SIZE proposals

- Lots of recent mindshare, I'll be brief
- Proposals to reduce, leave same, increase
- Pre-scheduled increase, such as BIP101 & others
- Miner collusion-determined limit
- flexcap (1, 2, 3, etc.)
- Penalties of: fees, subsidy, difficulty (like BIP105)
- Many other proposals...

Merged mining

- Use same hashrate for chance of mining (compatible) blocks on different chains
- Auxiliary and main chain are independent, both can have blocks unrelated to merged mining
- Use merkle tree to avoid including entire bitcoin transactions in auxiliary chain
- Auxiliary's validators OK with bitcoin block headers and blocks in auxiliary chain

https://en.bitcoin.it/wiki/Merged_mining_specification
http://bitcoin.stackexchange.com/questions/273/how-does-merged-mining-work

Fidelity-bonded ledgers

- Take out everything from bitcoin except transactions and scripting
- Receive a reward for providing a proof of double-spending (fidelity bonds)
- Commit fraud, lose the fidelity bond
- Chaum tokens

https://people.xiph.org/~greg/bitcoin-wizards-fraud-proof.log.txt
http://lists.linuxfoundation.org/pipermail/bitcoin-dev/2013-February/002189.html

Sidechains

- Sandboxed experimentation on alternate ledgers
- There's a paper and source code
- Mostly same scalability concerns, although some scale/security tradeoffs can be made for e.g. federated block signing...
- Lots of recent mind-share, so details skipped here

2-way peg and sidechains (diagram)

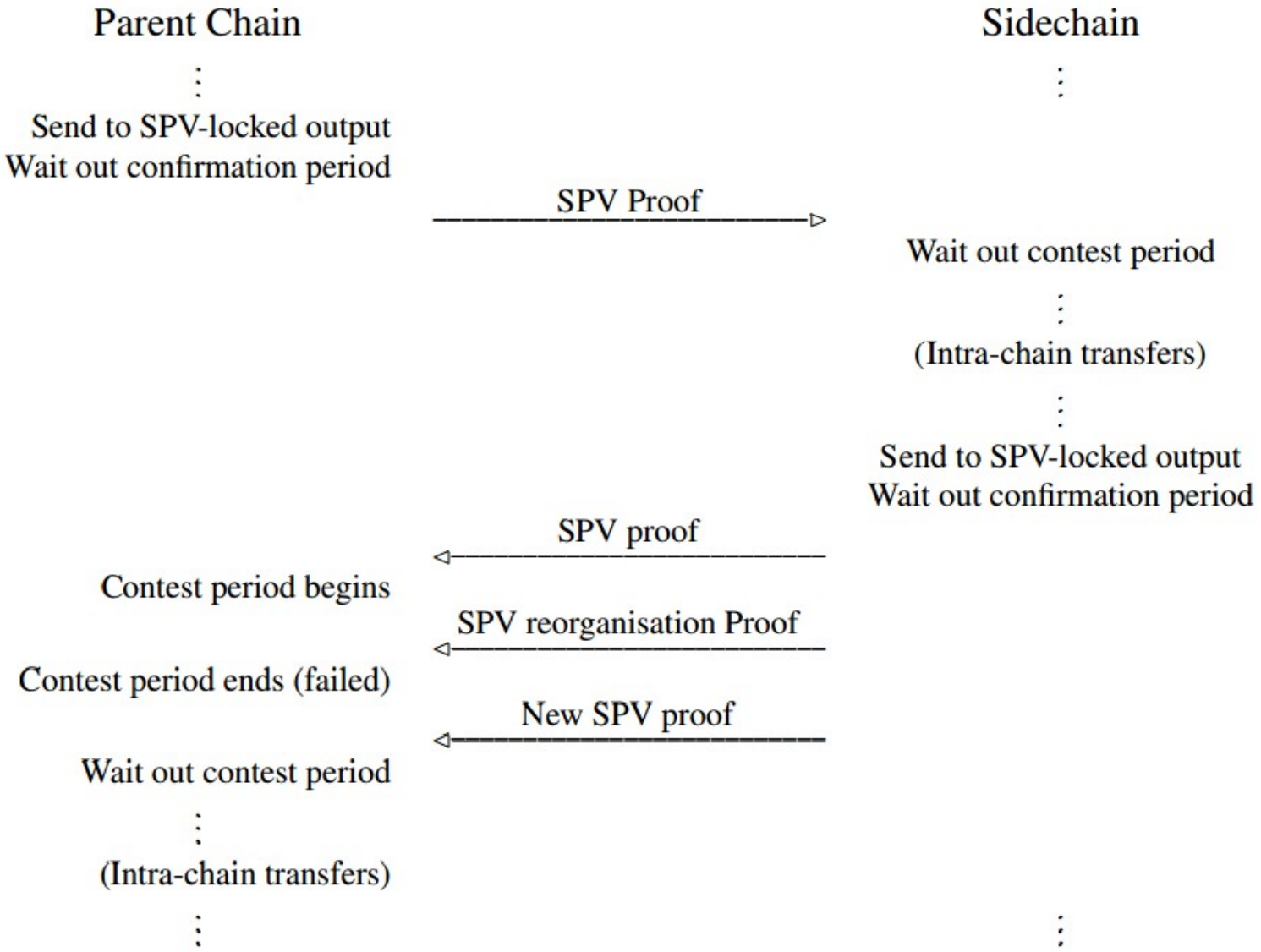


Figure 1: Example two-way peg protocol.

Extension blocks

- Block size proposal achievable with soft-fork
- Commitment to extension block put into main chain
- Users can opt-in to larger blocks by transferring BTC in and out of extension blocks
- Fee pressure differences because different security preferences and offerings
- New-version-only addresses maybe more secure here

https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2015-May/008356.html

Treechains

- Reduce blockchain to proof-of-publication mechanism
- Push all verification into the transaction receivers
- Miners don't verify except narrow timestamping rules, no digital signature verification etc.
- Wallets provide (perhaps large?) UTXOs and proofs
- As coin history is exponential in size, requires techniques such as recursive SNARKs or probabilistic verification to reduce coin validity proof size to O(n) or O(n log n).

http://lists.linuxfoundation.org/pipermail/bitcoin-dev/2014-March/004797.html

Atomic cross-chain swaps and 2-way pegs

- Trustless transfer of BTC to other ledgers
- Compact SPV proof for 2-way pegs
- Opt-in experimentation and opt-in risk
- Use contracts to coordinate mutual transfers on both sides
- Maybe cheaper to get new UTXOs on alternative ledgers or when participating as member of multisig pool?

https://en.bitcoin.it/wiki/Atomic_cross-chain_trading

Probabilistic payments

- Difference between payments vs. transactions
- One way: make signature commit to a hash (sign to contract)
- Join risk sharing pools
- Pre-bitcoin lottery micropayment schemes
- Various ways to mitigate double spending
- Other problems: people probably not happy about variance in salary/pay?

https://bitcointalk.org/index.php?topic=62558.0

PowPay

- Receiver is turned into mining pool
- Sender arranges PoW/hashrate as payment
- Privacy benefits
- Could encourage miner centralization
  - variance
  - overhead

http://lists.linuxfoundation.org/pipermail/bitcoin-dev/2013-May/002564.html

Cut-through

- Works for unconfirmed transactions only, fast confirmations means less time for cut-through to happen.
- Bob → 200 different Carols → same Alice
- Why keep Bob → Carol transactions? Unnecessary under cooperation.
- Circular flows in payment networks can also be (trustlessly) abbreviated or omitted from history.
- Use locktime to give time for summaries to be found
- Can individual nodes cooperate to find optimal transaction history abbreviations? better way?

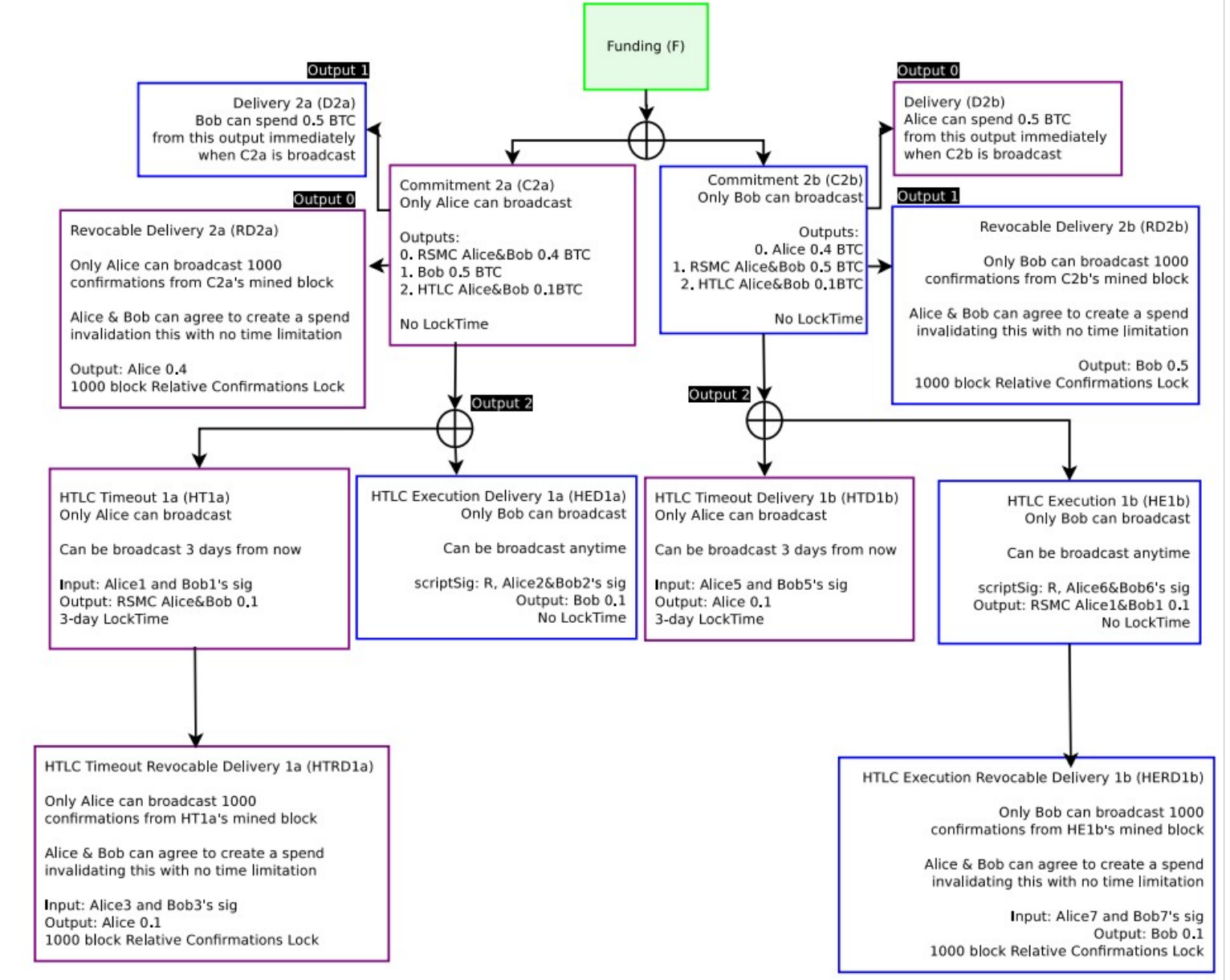
https://bitcointalk.org/index.php?topic=281848.0

Payment Channels and also Lightning Network

- Bi-directional payment channels
- Hub-and-spoke → multi-hop network
- Payment routing
- Channel liquidity, positive and negative fees
- Multi-chain UTXO ambivalence
- Requires channel setup, risk of worst-case channel closure delays dumping everything to blockchain.
- See also: Amiko Pay, Stroem/Strawpay

http://lightning.network/lightning-network-paper.pdf
https://github.com/ElementsProject/lightning

LN transaction diagram



LN abbreviations

- Revocable Sequence Maturity Contract (RSMC)
- Hashed Timelock Contract (HTLC)
- Commitment transaction
- Revocable delivery transaction
- Breach-remedy transaction
- HTLC Execution Delivery Transaction (HED)

Fraud proofs

- Compact proofs of rule violations, size scales with block size(?)
- Block headers could commit to both a valid and invalid block
- Requires fraud proofs (and fraud bounties?) of:
  - Invalid script
  - Double spending
  - False minting / spending non-existing input
  - False inflation (merkle sum tree of fees)
  - Oversized block (require all transactions)
  - Invalid signature (already supported)
  - Invalid UTXO commitments
- "It would be necessary to go through the entire set of consensus rules and create a fraud proof for every check that is performed."
- No good way to do fraud proof of censorship

https://bitcointalk.org/index.php?topic=1103281.msg11743498#msg11743498

Proof-of-Treachery

- 1 supernode miner or block signer
- Deterministic (obfuscated?) computation
- Provably-(in)valid state transitions
- How to recover consensus after supernode fraud?
  - How to recover consensus after mining cartel fraud?

https://en.bitcoin.it/wiki/User:Gmaxwell/features#Proofs

Fraud proofs (links)

- https://bitcointalk.org/index.php?topic=314506.0
- https://bitcointalk.org/index.php?topic=1103281.msg11743498#msg11743498
- https://bitcointalk.org/index.php?topic=137933.0
- https://github.com/TierNolan/bips/blob/9a8fac56c3817396910729c8c1fb3959686b301f/bip-sum-merk.mediawiki
- http://lists.linuxfoundation.org/pipermail/bitcoin-dev/2012-June/001632.html
- https://github.com/proofchains/python-proofchains

SNARKs (simplified)

- What if:
  - … you could prove F(x,y)=True for any program F
  - … without revealing y
  - … with a small constant-sized proof that verifies in milliseconds, no matter how complex F is
- Proofs of faithful execution
- Theorized since 80s but hasn't been practical
- Potentially secure somewhat practical construction found in past few years

SNARKs

- (Zero Knowledge) Succinct Non-interactive Arguments of Knowledge (zk-SNARKs)
- Proofs of Knowledge
- Check witness instead of executing the code yourself
- SNARKs offer a proof that code was faithfully executed
- This proof is sublinear size in the length of the execution, and can be verified in sublinear time.

https://tahoe-lafs.org/trac/tahoe-lafs/wiki/SNARKs
http://diyhpl.us/~bryan/papers2/bitcoin/snarks/

Zero Knowledge Validated History Replacements

- Create compact constant-sized proofs to show that a history replacement was the result of a faithful validation of the blockchain
- Could be used for/with pruned history proposals

Zero Knowledge Proof of Authorized UTXO Modification

- Mined blocks only provide updates to UTXO set
- Constant-sized proof that UTXO modification is an authorized modification
- Authorization derived from unspecified number of undisclosed transactions - blockchain doesn't need to store transactions ("One Big SNARK"), only proofs
- ECDSA verification in prover (or EC point addition?)
  - hash-based Lamport signatures could be verified much more quickly
- Send coins with (probably exponential in size) history
- Sublinear blockchain growth

SNARKs limitations

- Trusted setup
  - Versions without trusted setup are (so far) less efficient
- Very slow prover, on the order of a 10 Hz CPU
- Lots of new and untested cryptography, bitcoin mainnet is not the ideal testbed
- libsnark security hole
- May be more useful as a design tool for now
- We know this is all possible, but SNARKs are going to take a while

Other interesting directions

- SNARKs (libsnark, snarkfront), tinyram, oblivious RAM, etc.
- Publicly verifiable computation (VerSum, ...)
- Multi-party computation (MPC)
- Remote attestation
- Trusted setup vs. random oracle regimes
- Unexplored possibilities with exotic SIGHASH types and contracts
  - revocable delivery, breach-remedy, refunds, signed cascades prior to funding, …

This would be good

- UTXO commitments or similar
- Blockchain size goals:
  - Sublinear size growth
  - Constant size, use pruning
- Wallets provide necessary UTXOs and proof
- … without trusted setup.

Useful principles

- "We don't care what the history is, just that it doesn't change."
- Input and output amounts need to be conserved, but unknown (to us) is OK

"""

every advanced crypto concept is just

a) complicated thing

b) compicated thing

c) merkle tree on top of complicated things

d) complicated thing

"""

Tracking bitcoin tech inventions

- Fallout of looking at many previous proposals
- Mostly bitcoin tech proposals, inventions
- Significantly less formal than BIPs
- >800 tagged jotmuch bookmarks
- YAML available upon request
- Will be using a git (wiki) repo

http://diyhpl.us/~bryan/irc/bitcoin/bitcoin-selected-bookmarks.2015-09-09.txt

Review of Bitcoin Scaling Proposals

These slides can be found on the web:

http://diyhpl.us/~bryan/irc/bitcoin/scalingbitcoin-review.pdf