Extensibility, Fraud Proofs, and Security Models

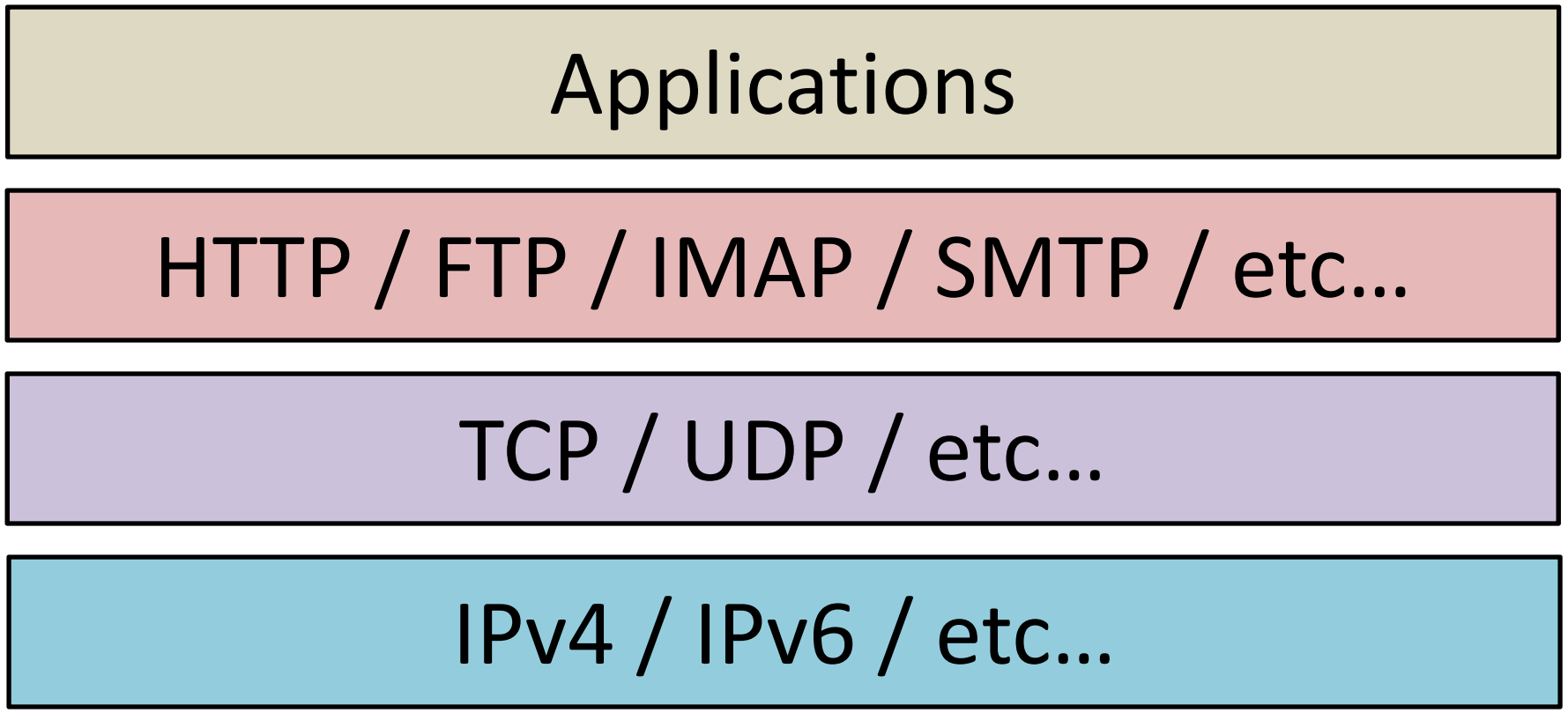
Eric Lombrozo
Founder, Co-CEO & Chief Technology Officer
Ciphrex Corp.

email:	eric@ciphrex.com
skype:	eric.lombrozo
irc:	CodeShark

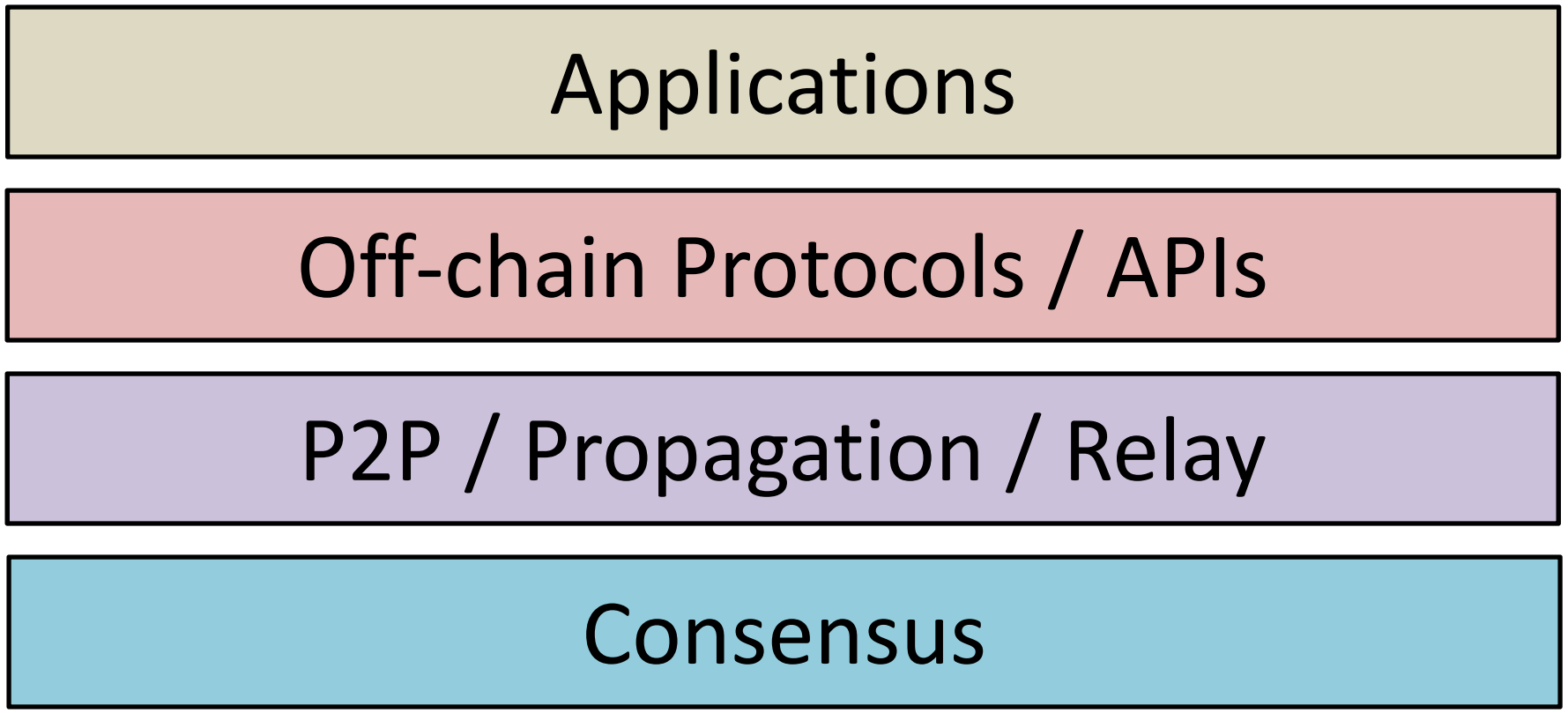
SCALING BITCOIN, Hong Kong – December 7, 2015

Why is Bitcoin so hard to change?

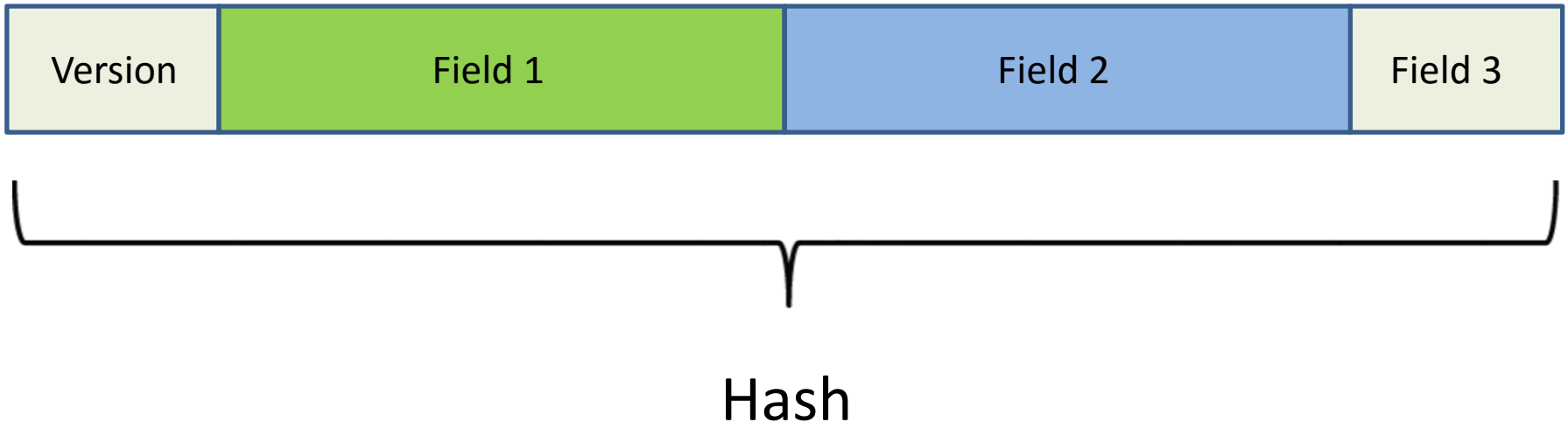
Internet Protocol Layers



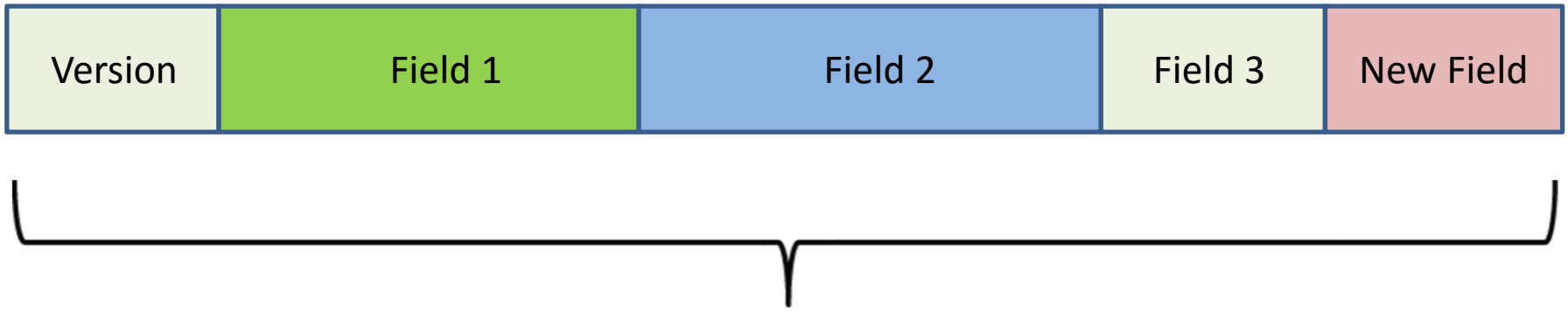
Bitcoin Protocol Layers



Serialization & Hashes



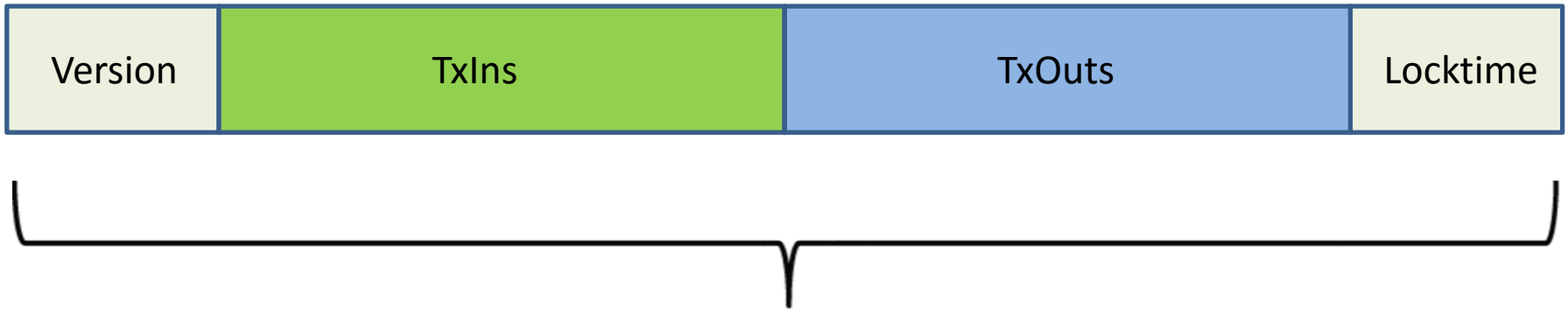
Serialization & Hashes



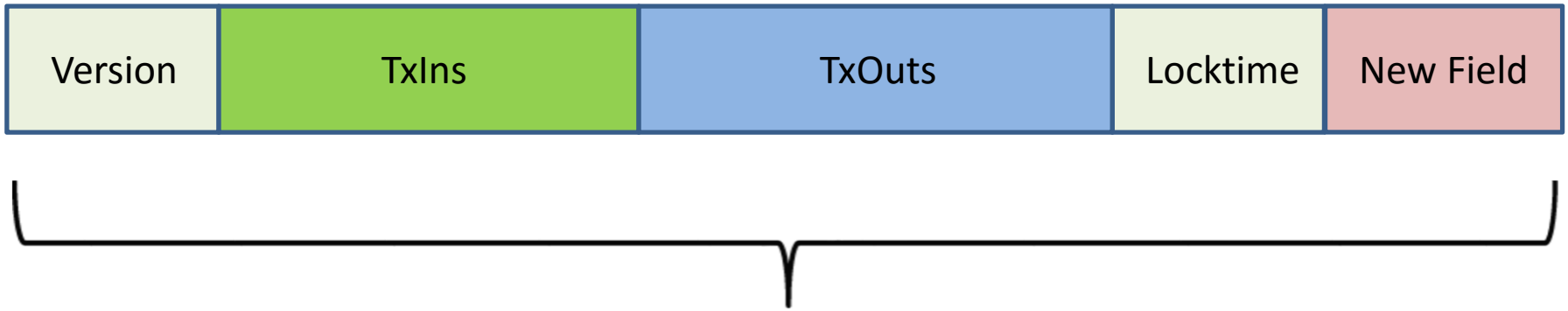
How can we
extend structures?

Layer Separation: Segregated Witness

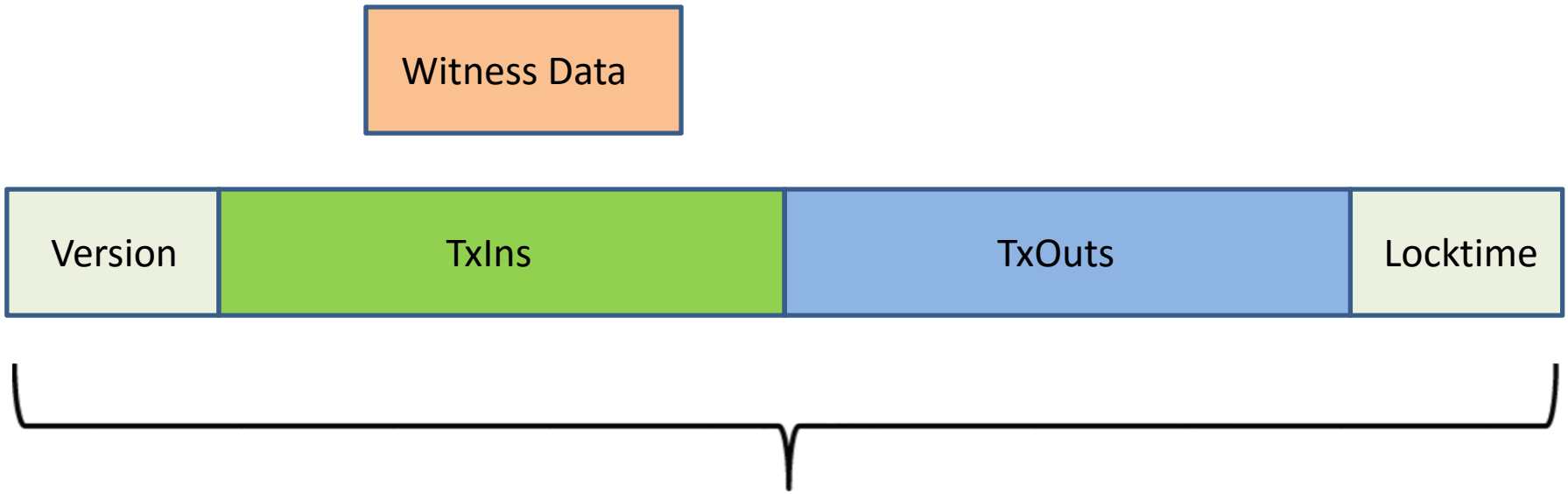
Transaction



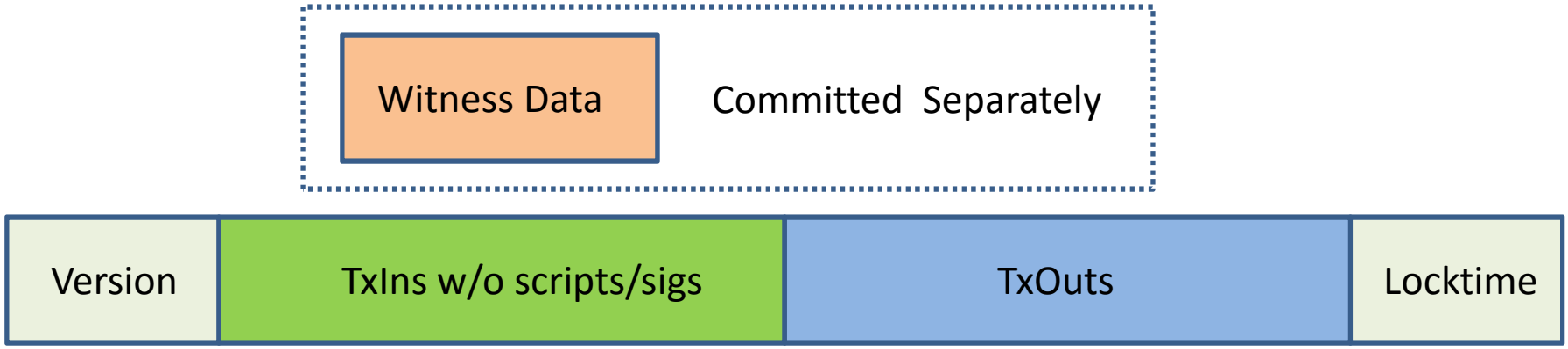
Transaction



Transaction



Consensus Layer



P2P Layer

Relayed Together

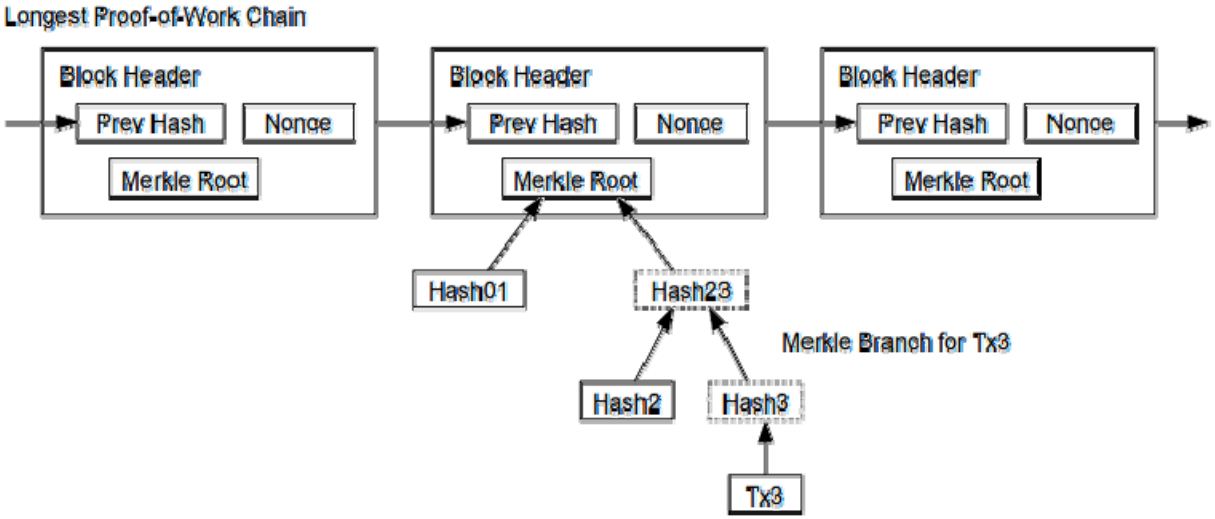
Application:
Fraud Proofs

SECURITY MODELS

- the Bobchain
- Full validation
- Nonvalidated SPV
- Partial validation / fraud proofs

8. Simplified Payment Verification

It is possible to verify payments without running a full network node. A user only needs to keep a copy of the block headers of the longest proof-of-work chain, which he can get by querying network nodes until he's convinced he has the longest chain, and obtain the Merkle branch linking the transaction to the block it's timestamped in. He can't check the transaction for himself, but by linking it to a place in the chain, he can see that a network node has accepted it, and blocks added after it further confirm the network has accepted it.

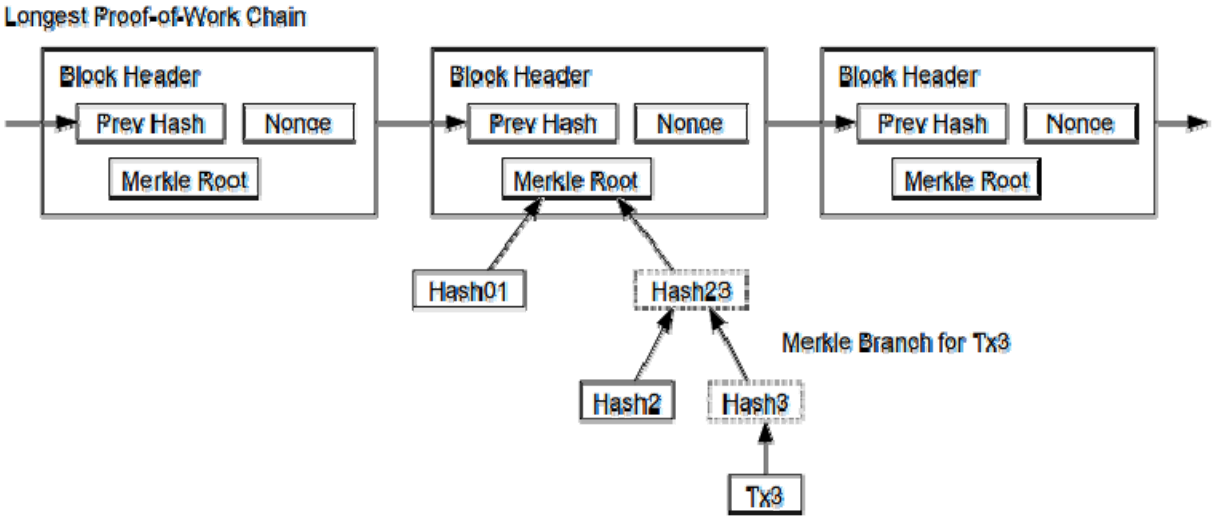


As such, the verification is reliable as long as honest nodes control the network, but is more vulnerable if the network is overpowered by an attacker. While network nodes can verify transactions for themselves, the simplified method can be fooled by an attacker's fabricated transactions for as long as the attacker can continue to overpower the network. One strategy to protect against this would be to accept alerts from network nodes when they detect an invalid block, prompting the user's software to download the full block and alerted transactions to confirm the inconsistency. Businesses that receive frequent payments will probably still want to run their own nodes for more independent security and quicker verification.

S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," https://bitcoin.org/bitcoin.pdf, 2008.

8. Simplified Payment Verification

It is possible to verify payments without running a full network node. A user only needs to keep a copy of the block headers of the longest proof-of-work chain, which he can get by querying network nodes until he's convinced he has the longest chain, and obtain the Merkle branch linking the transaction to the block it's timestamped in. He can't check the transaction for himself, but by linking it to a place in the chain, he can see that a network node has accepted it, and blocks added after it further confirm the network has accepted it.



As such, the verification is reliable as long as honest nodes control the network, but is more vulnerable if the network is overpowered by an attacker. While network nodes can verify transactions for themselves, the simplified method can be fooled by an attacker's fabricated transactions for as long as the attacker can continue to overpower the network. One strategy to protect against this would be to accept alerts from network nodes when they detect an invalid block, prompting the user's software to download the full block and alerted transactions to confirm the inconsistency. Businesses that receive frequent payments will probably still want to run their own nodes for more independent security and quicker verification.

S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," https://bitcoin.org/bitcoin.pdf, 2008.

Fraud Proofs

- Limits (size, op count)
- Conservation (inflation, fees)
- Existence (double-spending, false minting)
- Crypto (hashes, signatures)

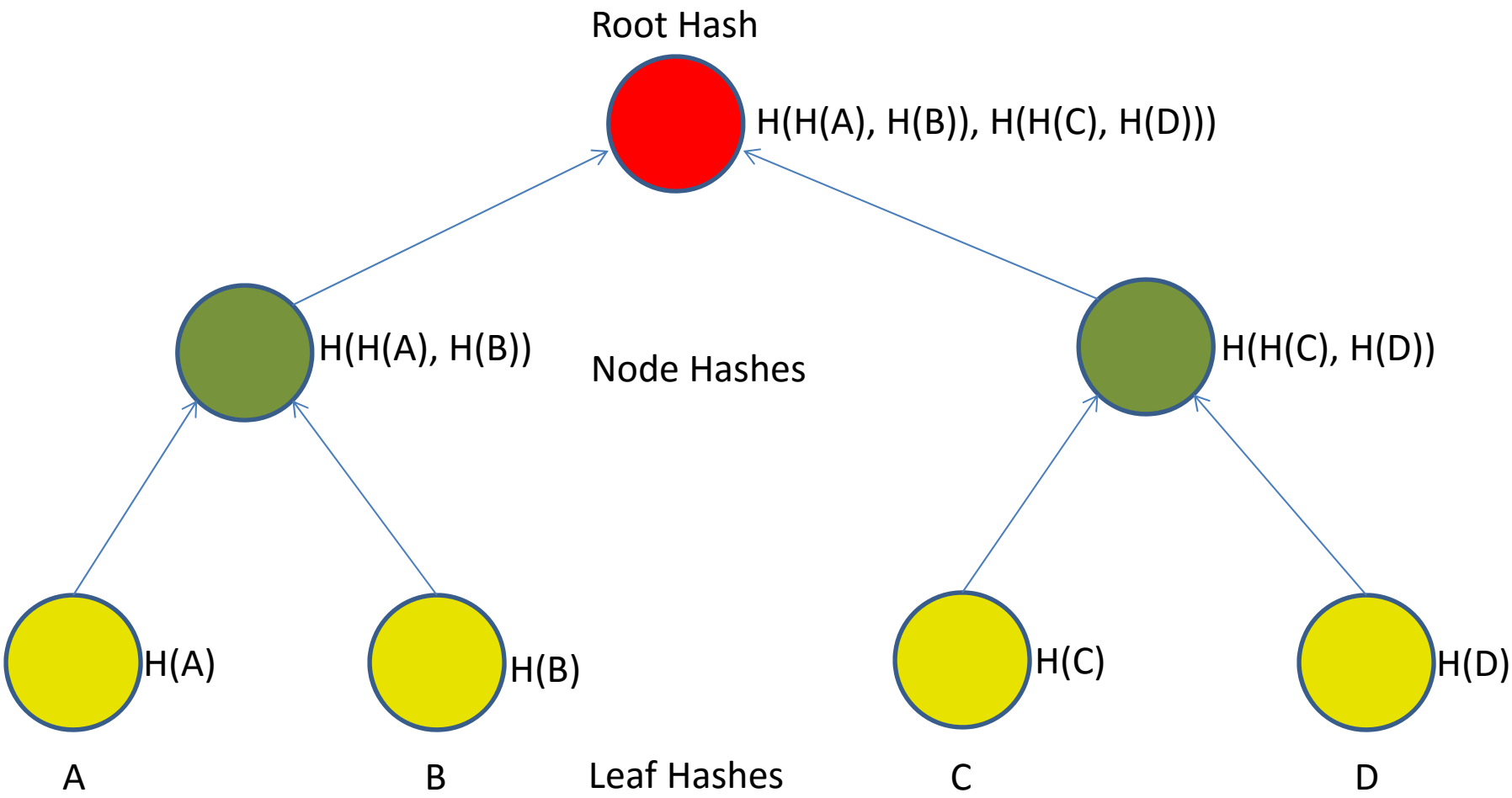
Fraud Proofs:
New Assumptions

- Strong relay censorship resistance
- Incentives for creating & propagating fraud proofs

Proofs & Validation

- Proof-of-work
- Hash Trees

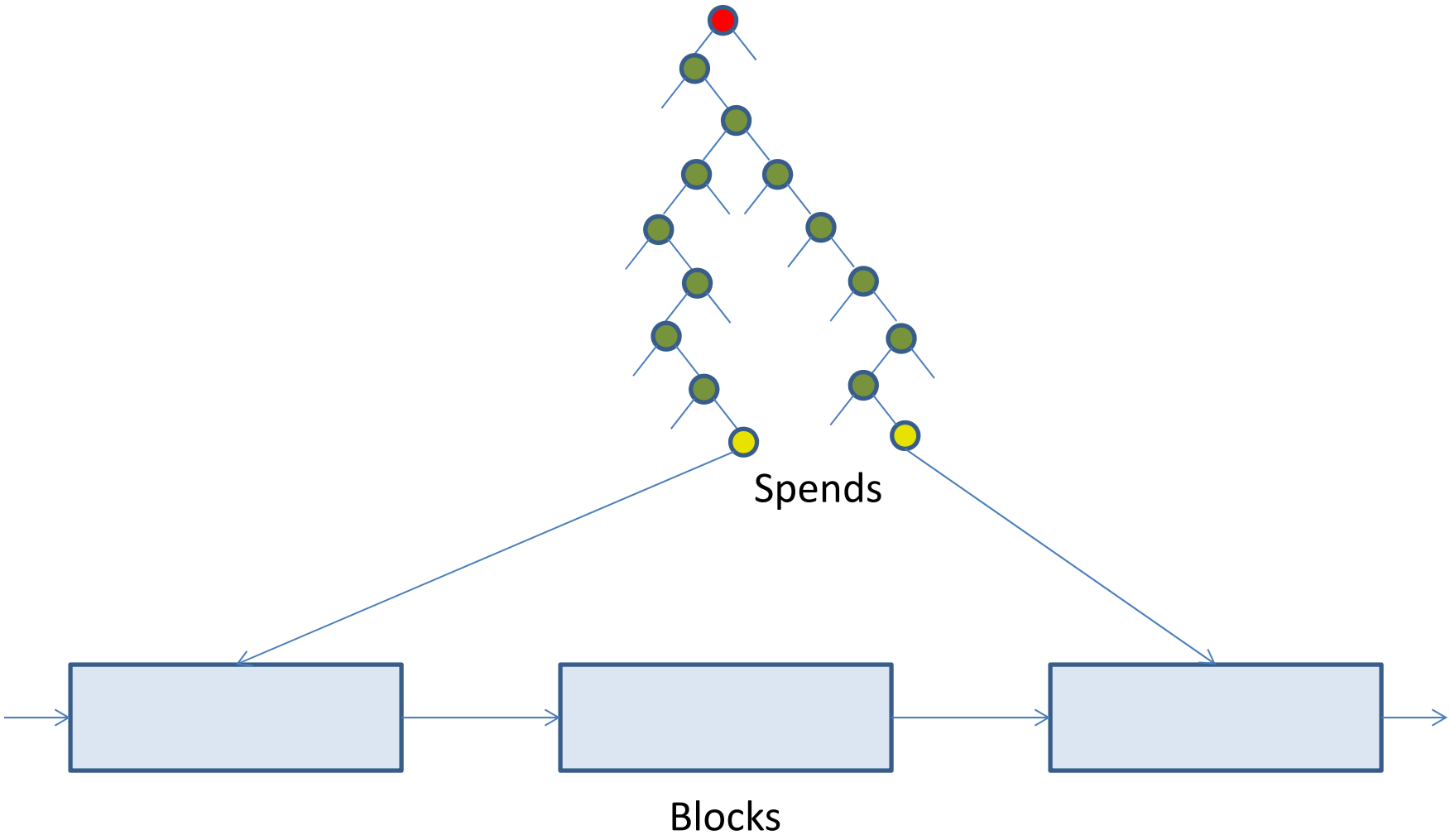
Hash Trees


Root Hash
H(H(A), H(B)), H(H(C), H(D)))
H(H(A), H(B))
Node Hashes
H(H(C), H(D))
H(A)
H(B)
H(C)
H(D)
A
B
Leaf Hashes
C
D

Proofs & Validation

- Proof-of-work
- Hash Trees
- Back References

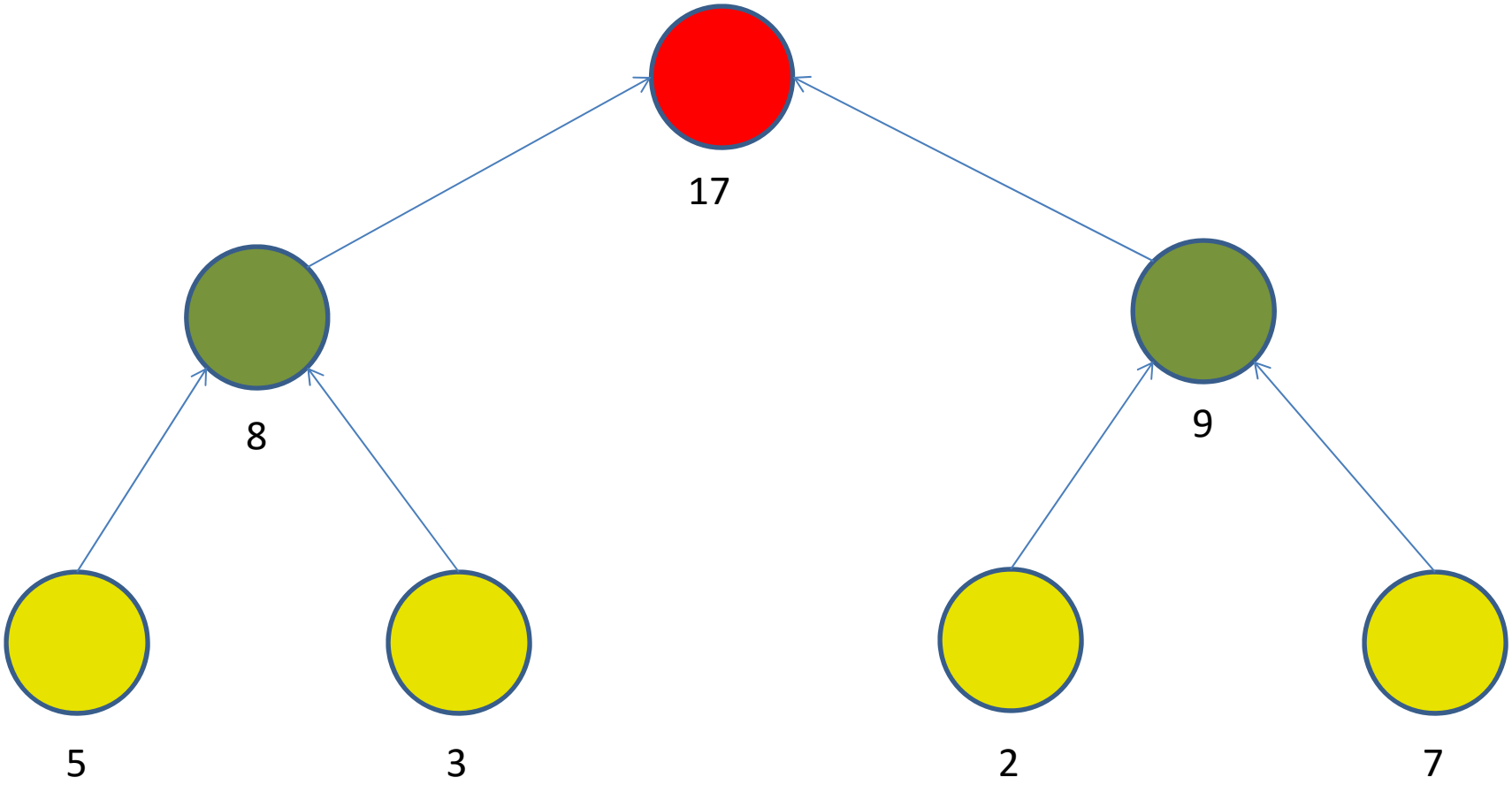
Back References



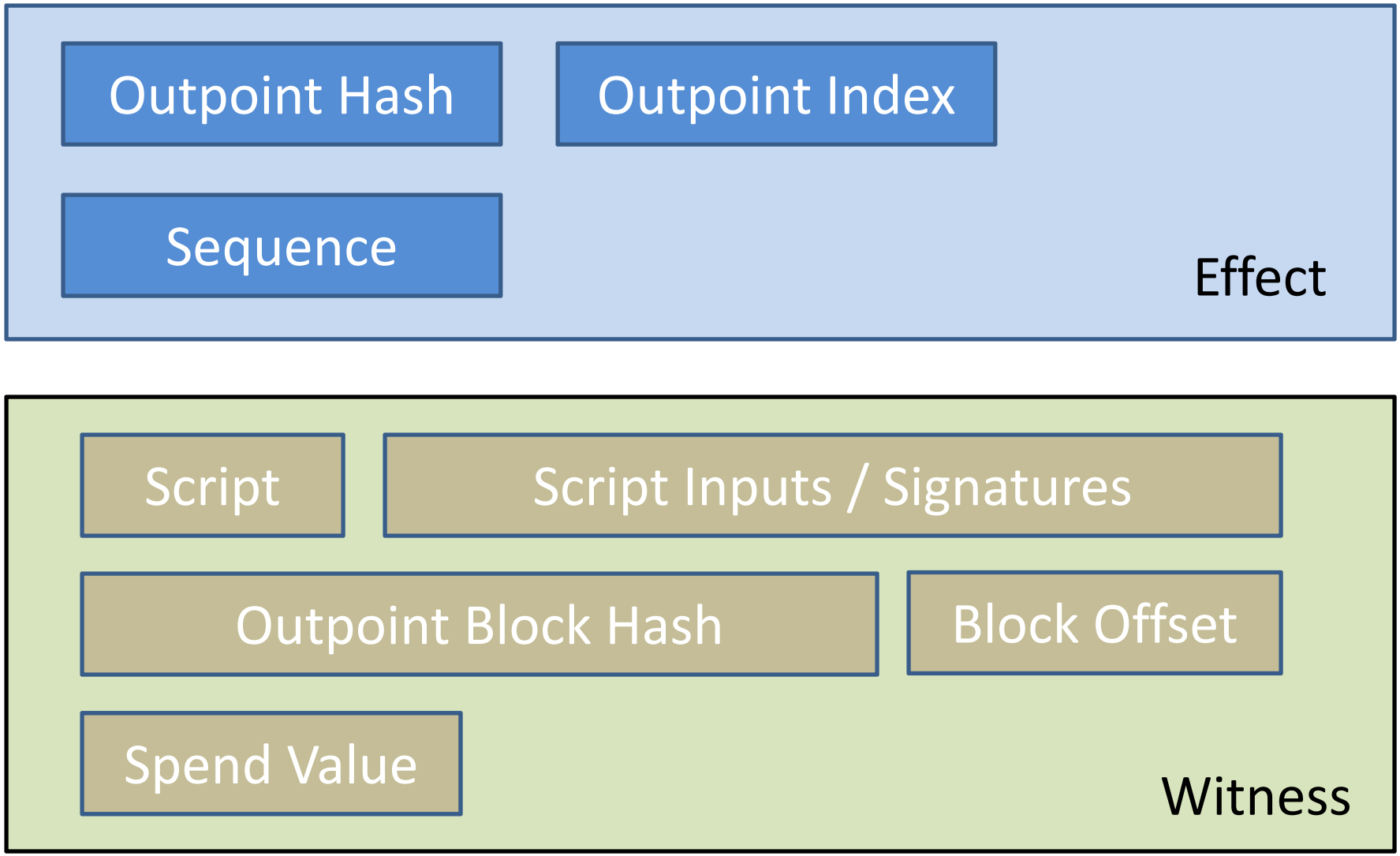
Proofs & Validation

- Proof-of-work
- Hash Trees
- Back References
- Sum Trees

Sum Trees



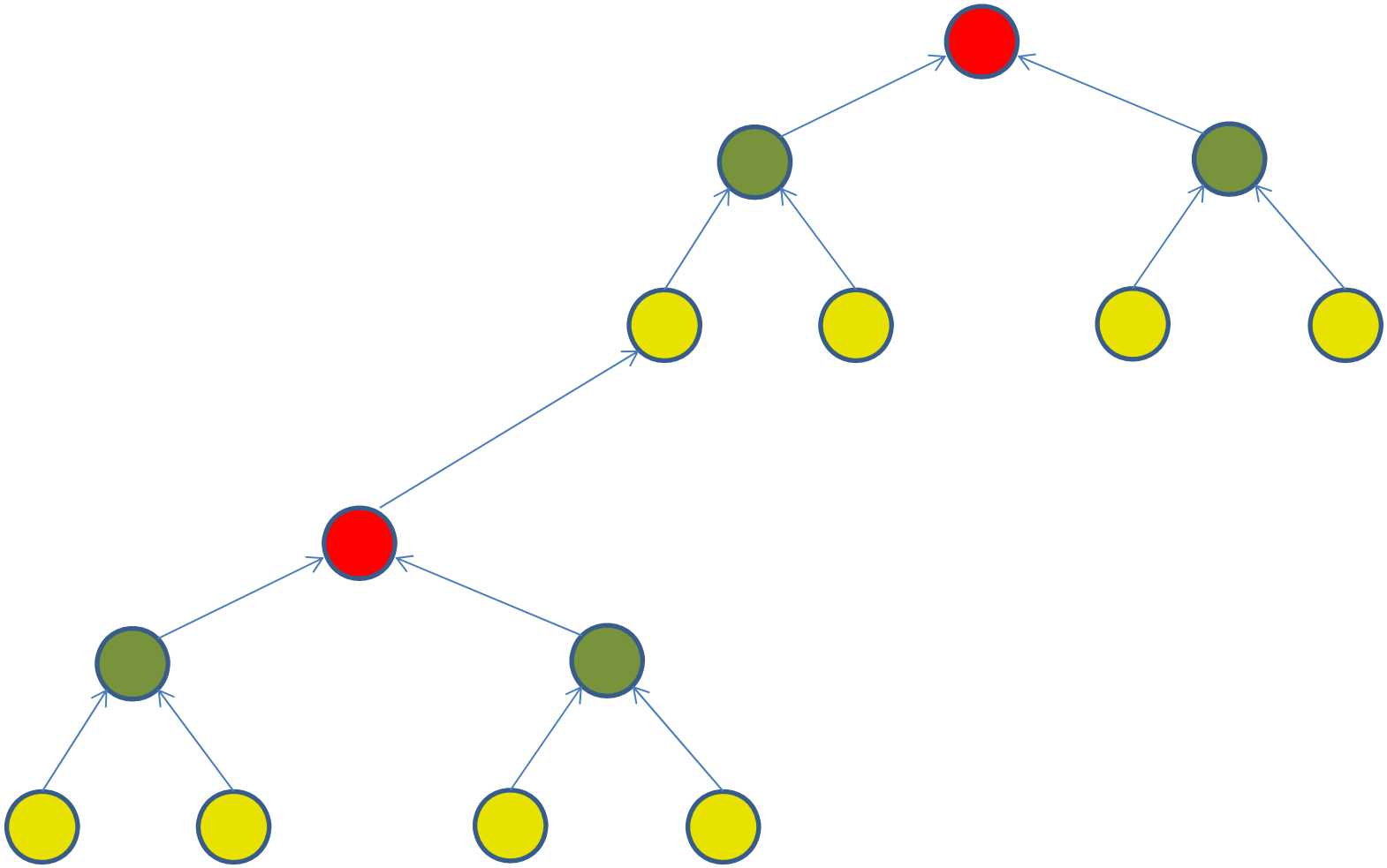
SegWit TxIn Structure



The Future…

- Feature addition soft forks
  (tricky but has been done successfully)

Soft Fork Commitment Nesting

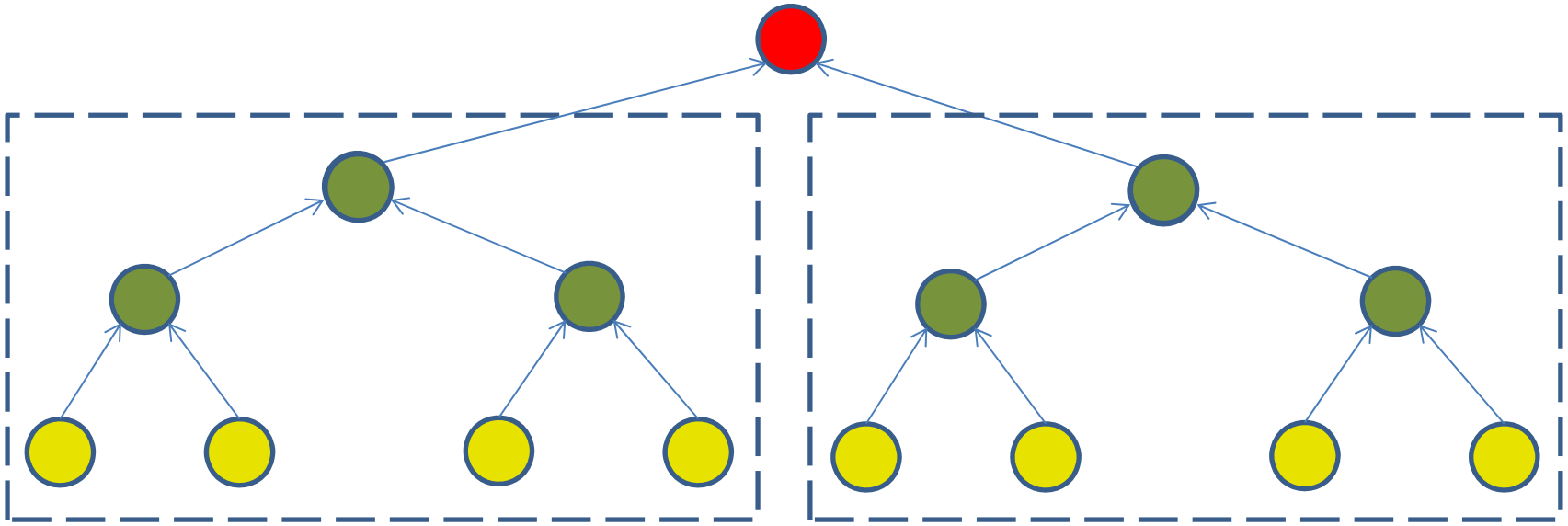


The Future…

- Feature addition soft forks
  (tricky but has been done successfully)

- Optimization hard forks
  (technically tricky, politically simple-ish)

Hard Fork Commitment Cleanup



Old Commitments

New Commitments

The Future…

- Feature addition soft forks
  (tricky but has been done successfully)
- Optimization hard forks
  (technically tricky, politically simple-ish)
- Contentious hard forks
  (VERY HARD…but probably necessary at some point)

Thank You!

email: eric@ciphrex.com

skype: eric.lombrozo

irc: CodeShark

Special thanks to Pieter Wuille, Gregory Maxwell, and Luke-Jr