Blockchains Meet Distributed Hash Tables:
Decoupling Validation from State Storage

Matteo Bernardini, Diego Pennino, and Maurizio Pizzonia

The problem

The problem

Scalability

The problem

Scalability

Consensus

The problem

Scalability

Consensus

Storage

Problems:

The problem

Scalability

Consensus

Storage

Problems:

- *Time of Synchronization*

The problem

Scalability

Consensus

Storage

Problems:

- *Time of Synchronization*
- *Storage capacity*

The problem

Scalability

Consensus

Storage

Problems:

- *Time of Synchronization*
- *Storage capacity*
- *Validation needs Storage*

Overview of our work

Full Node divided into two roles:

Overview of our work

Full Node divided into two roles:

1. **Storage role**:
   - ✓ tunable occupied space

Main idea:
a partitioned Ledger

Overview of our work

Full Node divided into two roles:

1. **Storage role**:
   - ✓ tunable occupied space

Main idea:
a partitioned Ledger

2. **Validation role**:
   - ✓ small chain with a constant size

Main idea:
validation from a secure starting point which **IS NOT** the local ledger

Storage Role

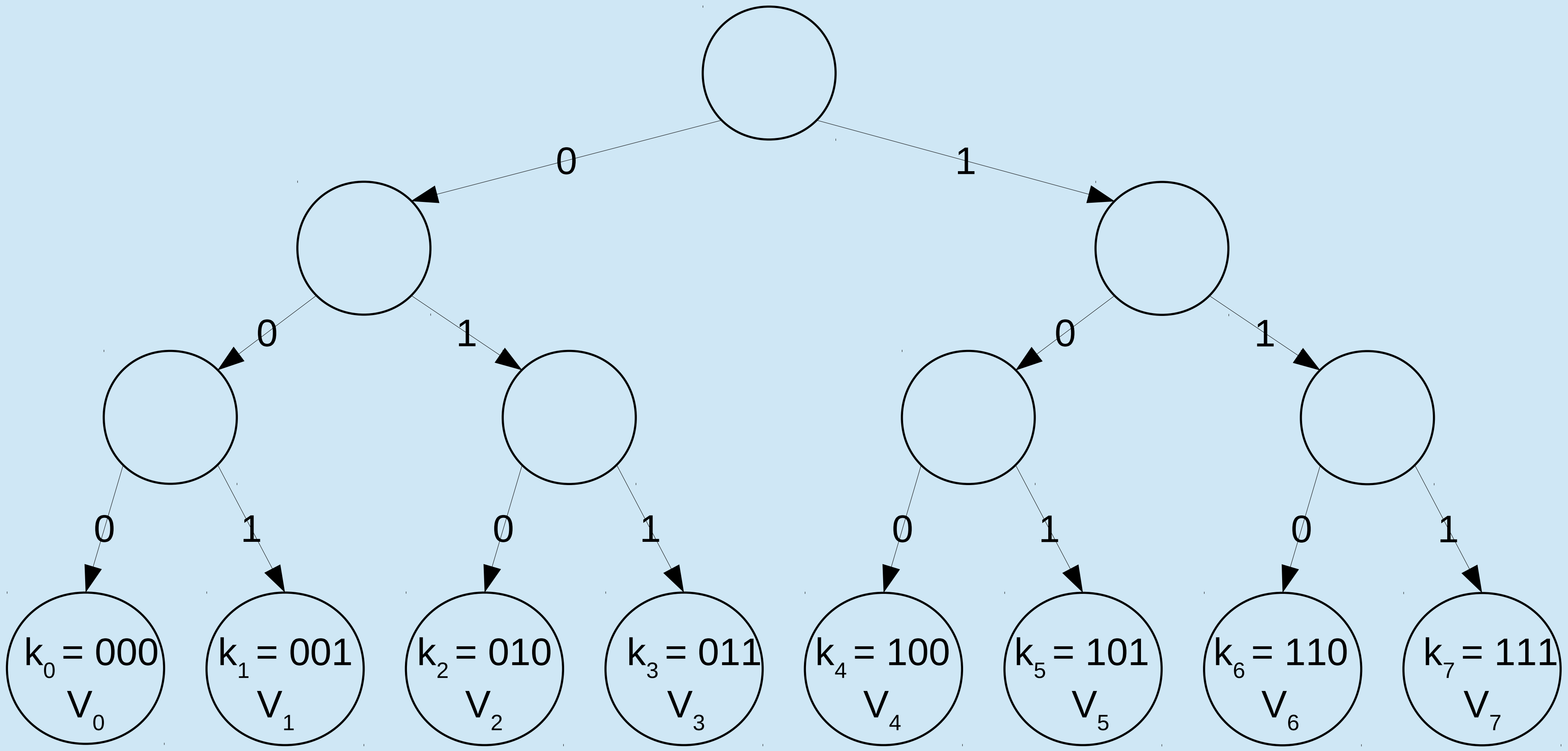
Ethereum Storage Role: *the Ledger*

k_0 = 000	k_1 = 001	k_2 = 010	k_3 = 011	k_4 = 100	k_5 = 101	k_6 = 110	k_7 = 111
V_0	V_1	V_2	V_3	V_4	V_5	V_6	V_7

State elements

Ethereum Storage Role: *the Ledger*

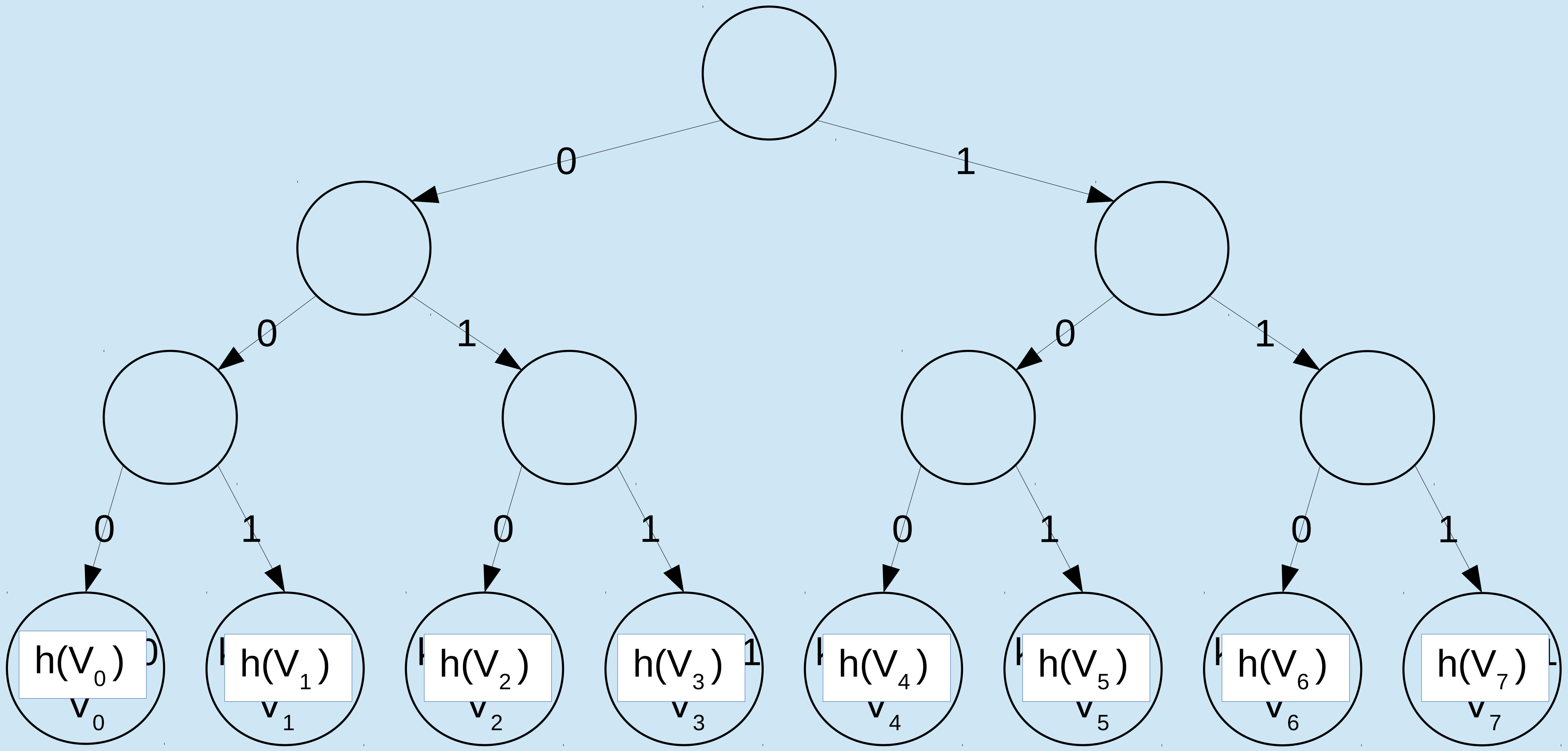
Prefix tree



State elements

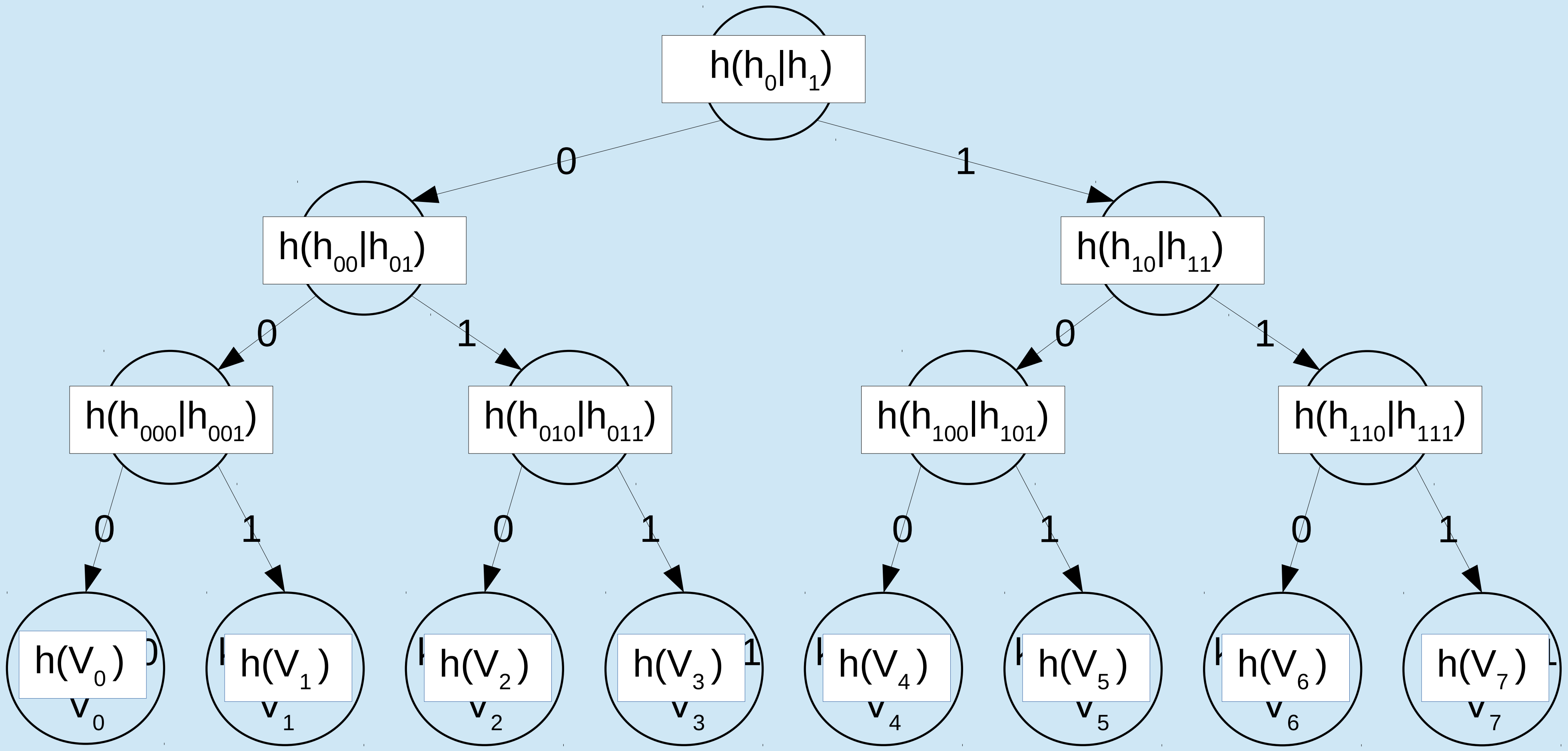
Ethereum Storage Role: *the Ledger*

*Prefix tree - **Merkle Hash Tree** (MHT)*



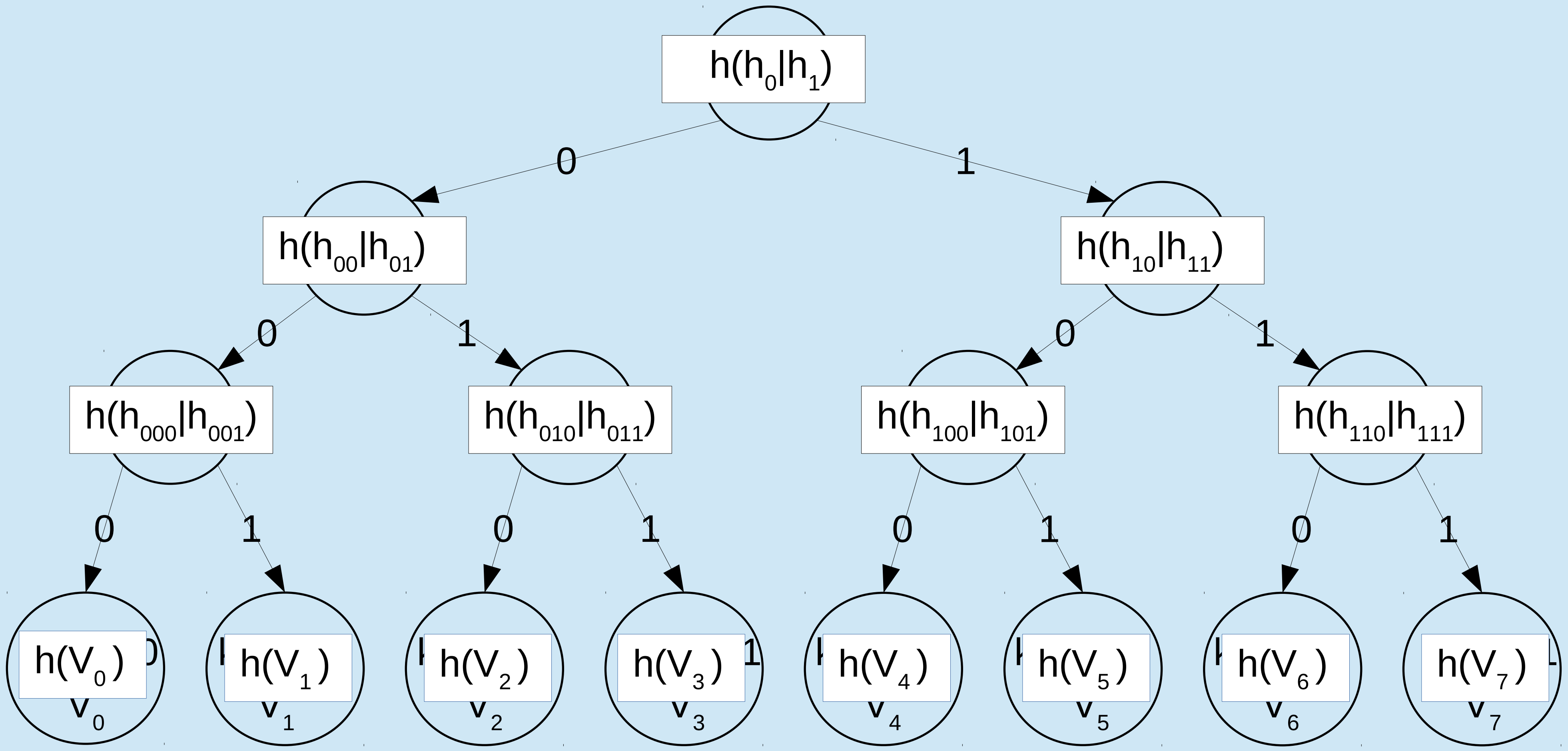
Ethereum Storage Role: *the Ledger*

*Prefix tree - **Merkle Hash Tree** (MHT)*



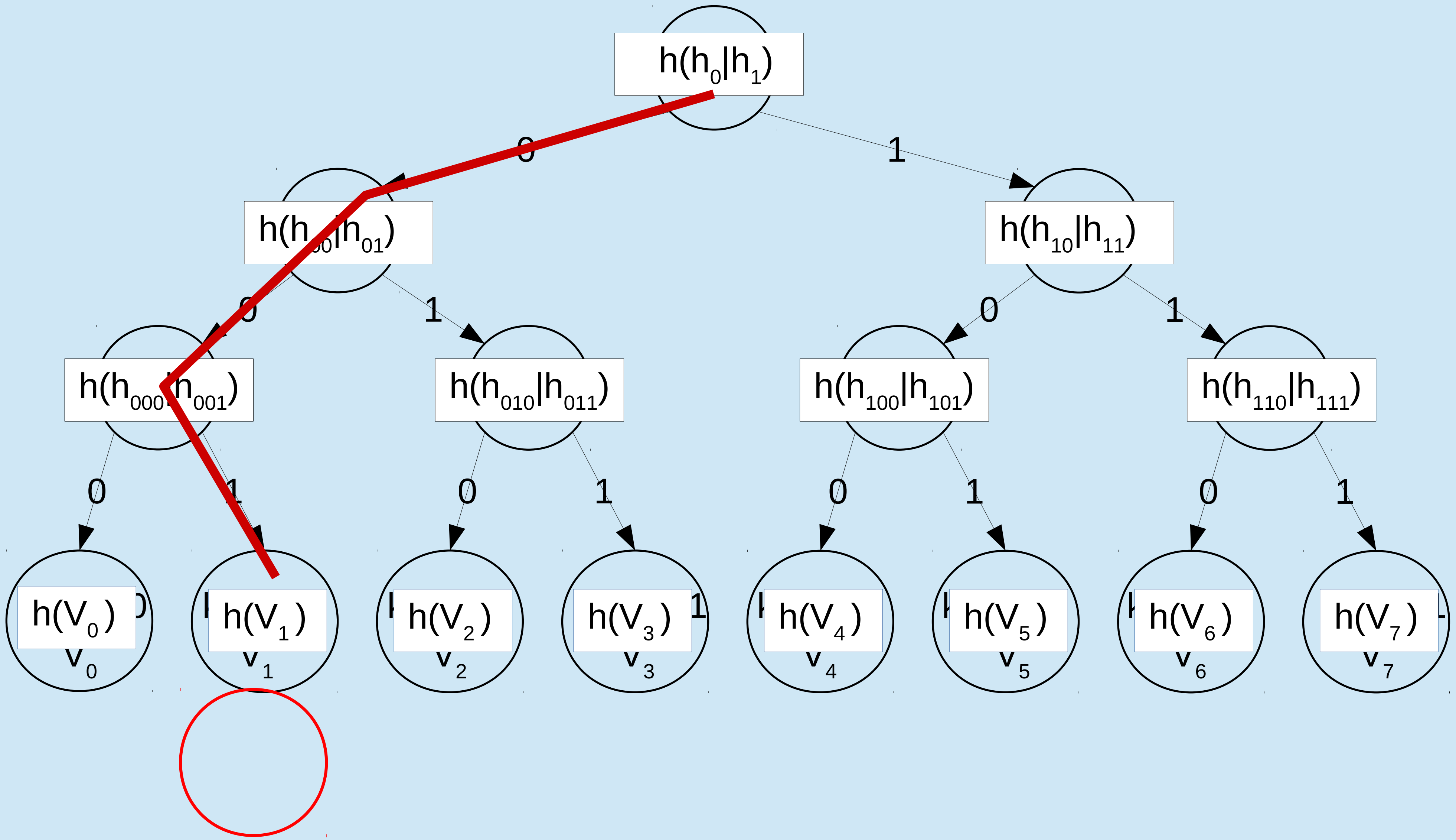
Ethereum Storage Role: *the Ledger*

MHT *- the proof*



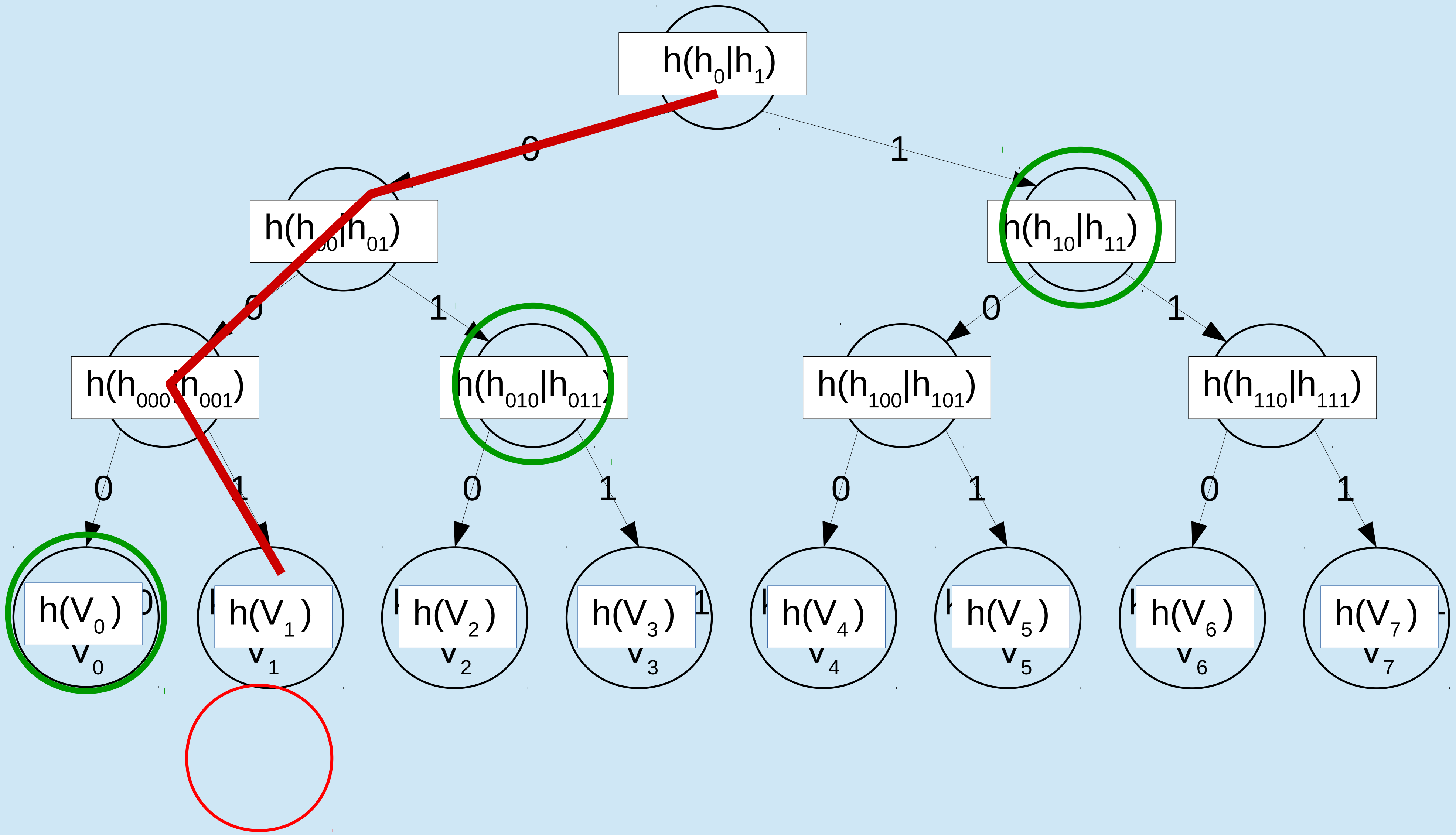
Ethereum Storage Role: *the Ledger*

MHT *- the proof*



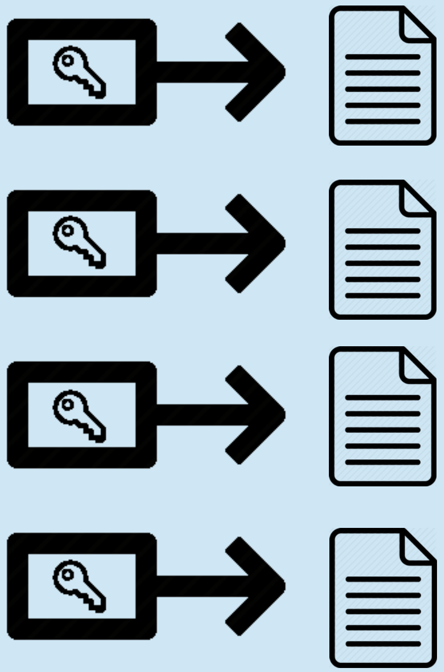
Ethereum Storage Role: *the Ledger*

MHT *- the proof*



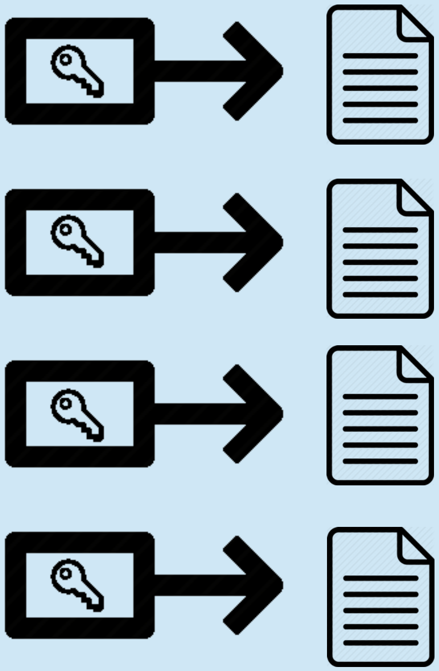
Storage Role of a Node: **Distributed Hash Table (DHT)**

Hash table (key-value)



Storage Role of a Node: **Distributed Hash Table (DHT)**

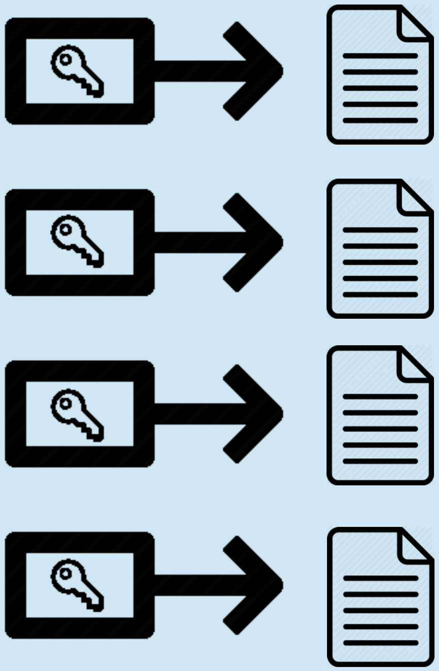
Hash table (key-value)



P2P

Storage Role of a Node: **Distributed Hash Table (DHT)**

Hash table (key-value)

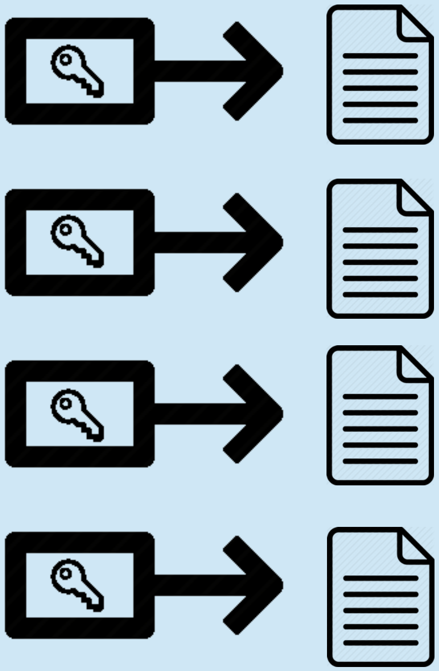


P2P

Authority of a key = it stores its value

Storage Role of a Node: **Distributed Hash Table (DHT)**

Hash table (key-value)



Primitives:

- *put*(k,v)
- *get*(k)

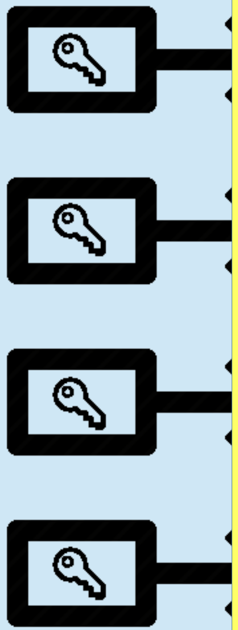
P2P

Authority of a key = it stores its value

Storage Role of a Node: **Distributed Hash Table (DHT)**

Hash table (key-value)

P2P



Properties:

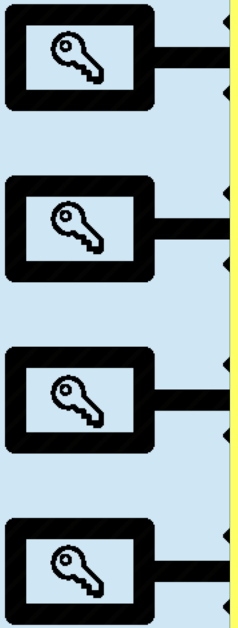
Primitives:
- *put*(k,v)
- *get*(k)

Authority of a key = it stores its value

Storage Role of a Node: **Distributed Hash Table (DHT)**

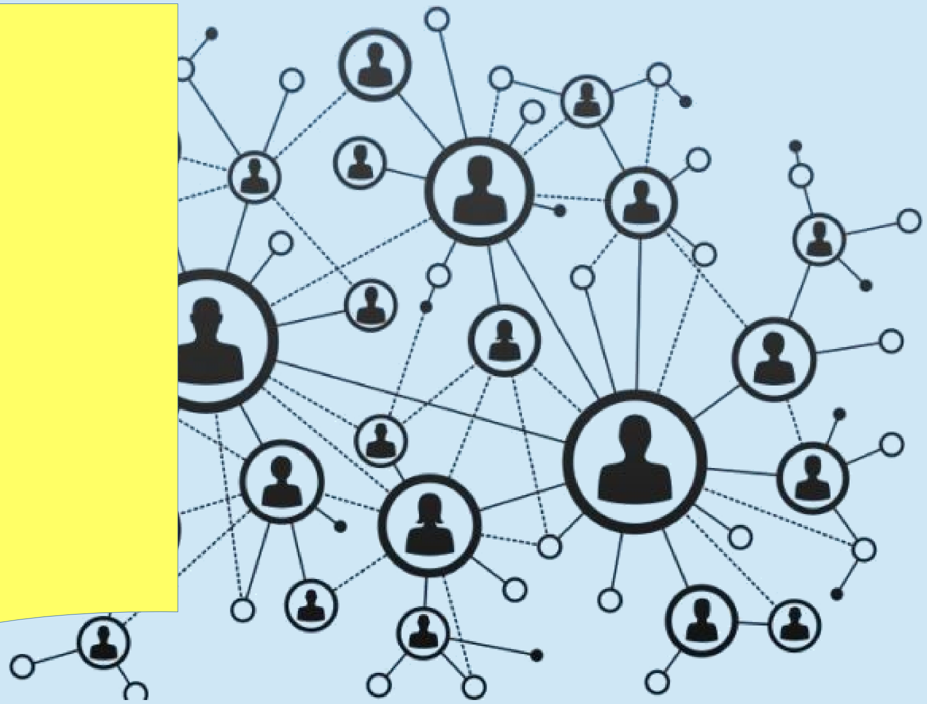
Hash table (key-value)

P2P



Properties:

- Autonomy and decentralization



Primitives:

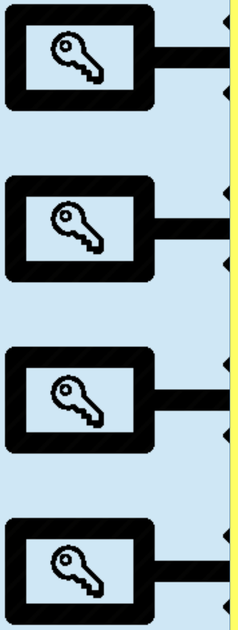
- *put*(k,v)
- *get*(k)

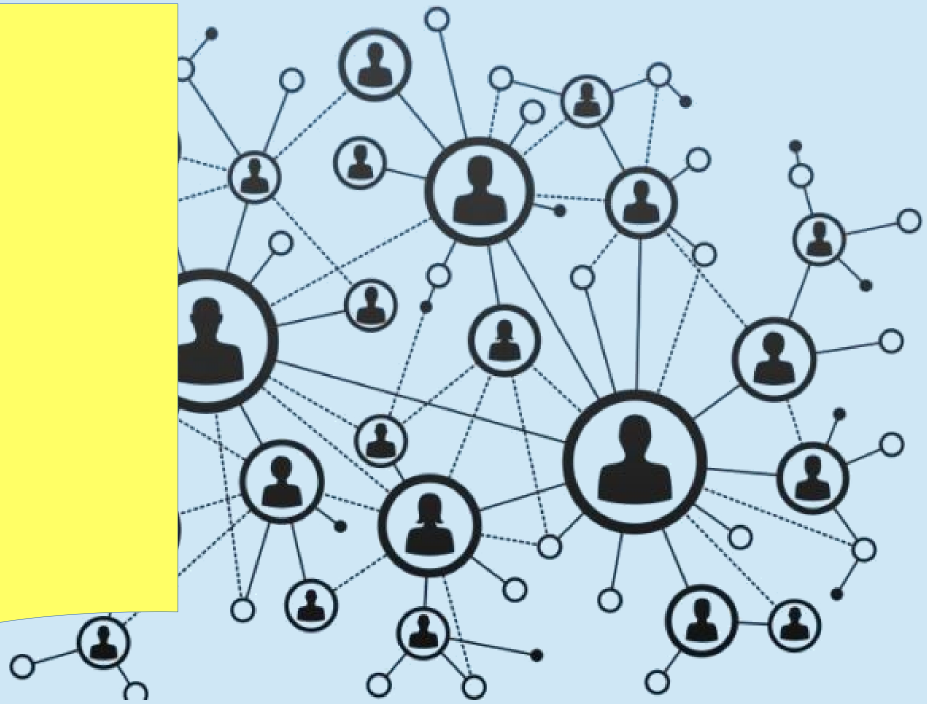
Authority of a key = it stores its value

Storage Role of a Node: **Distributed Hash Table (DHT)**

Hash table (key-value)

P2P





Properties:

- Autonomy and decentralization
- Fault tolerance

Primitives:

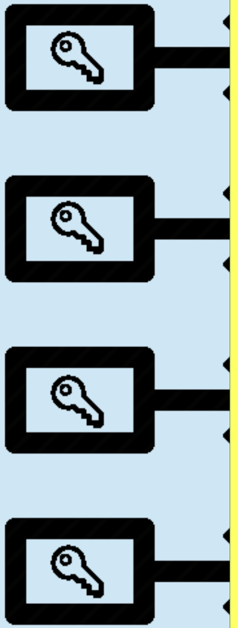
- *put*(k,v)
- *get*(k)

Authority of a key = it stores its value

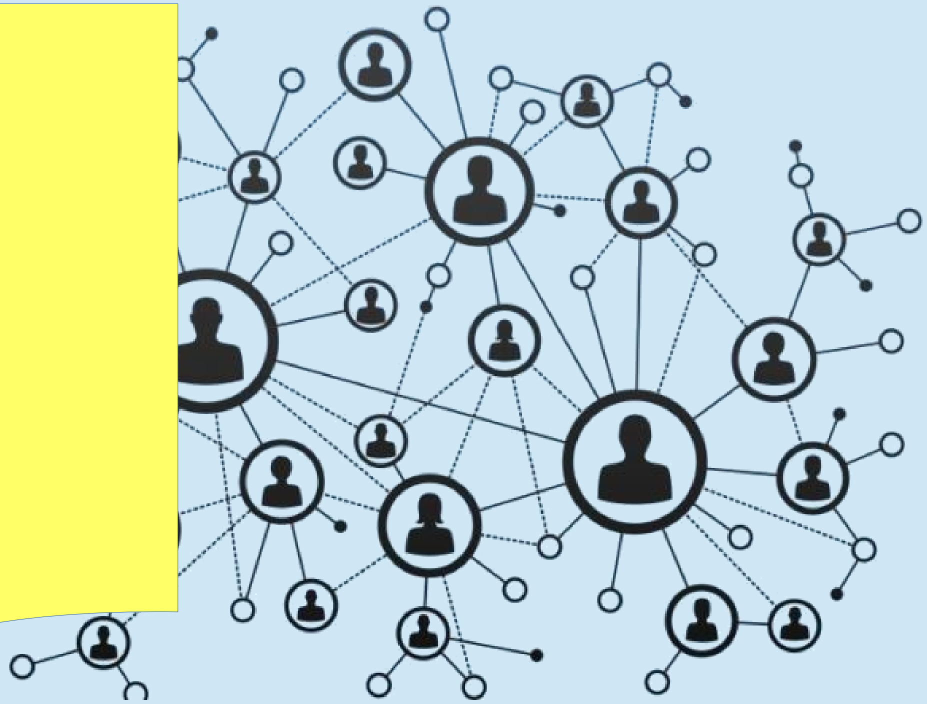
Storage Role of a Node: **Distributed Hash Table (DHT)**

Hash table (key-value)

P2P



Properties:
- Autonomy and decentralization
- Fault tolerance
- Scalability

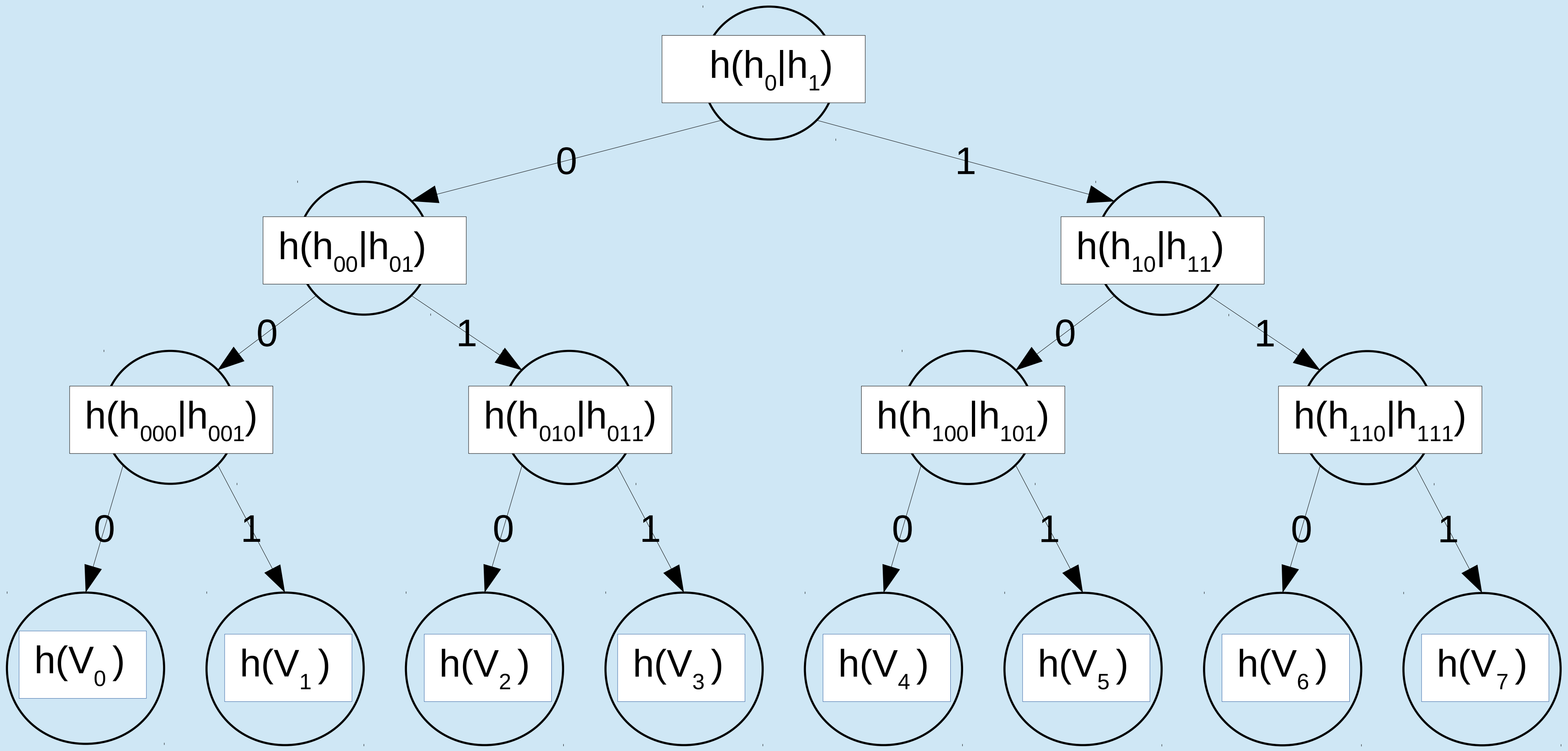


Primitives:
- *put*(k,v)
- *get*(k)

Authority of a key = it stores its value

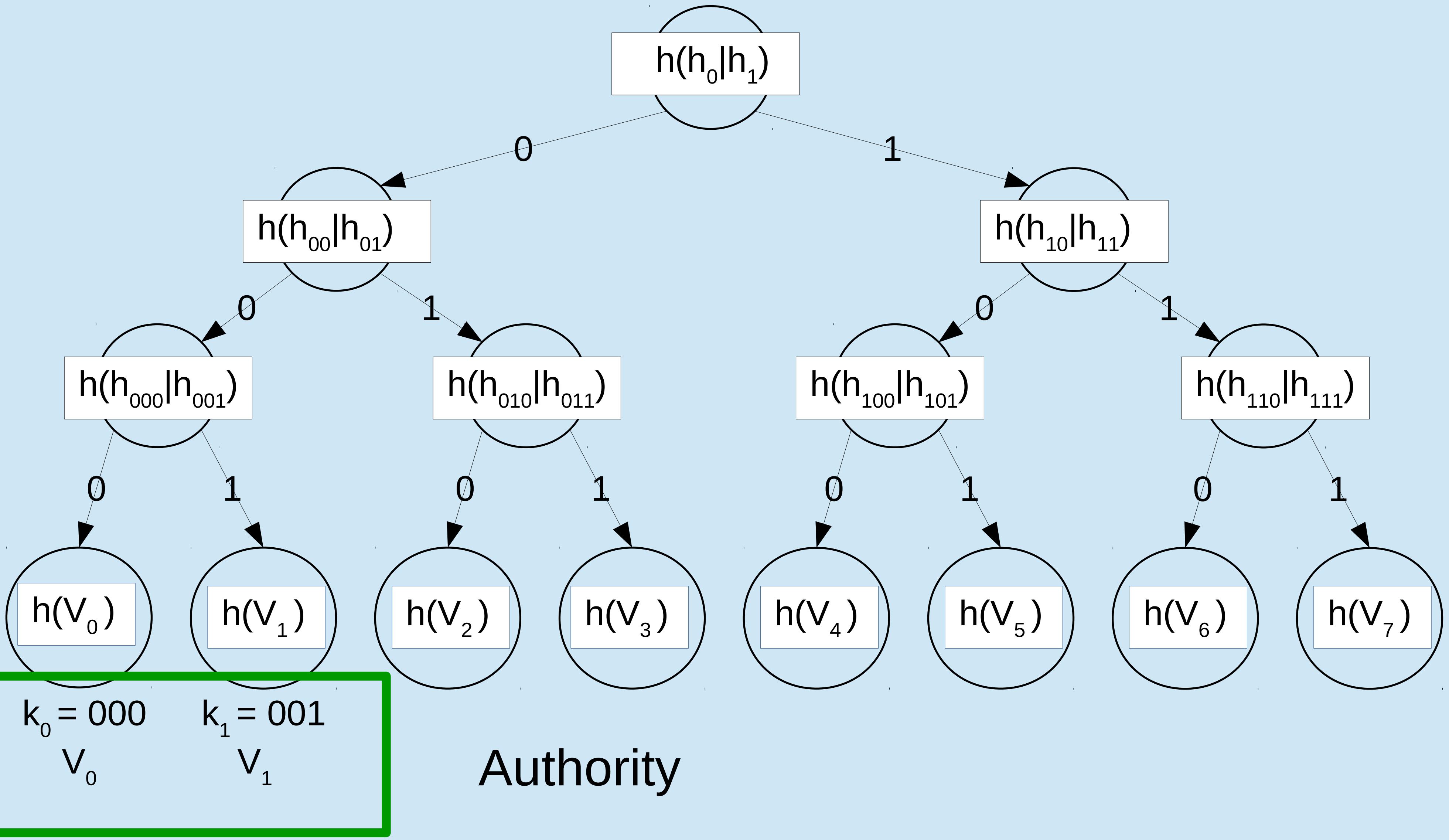
OUR Storage Role: *the small Ledger*

MHT + DHT = ***pruned Authenticated Data Structure*** *(pADS)*



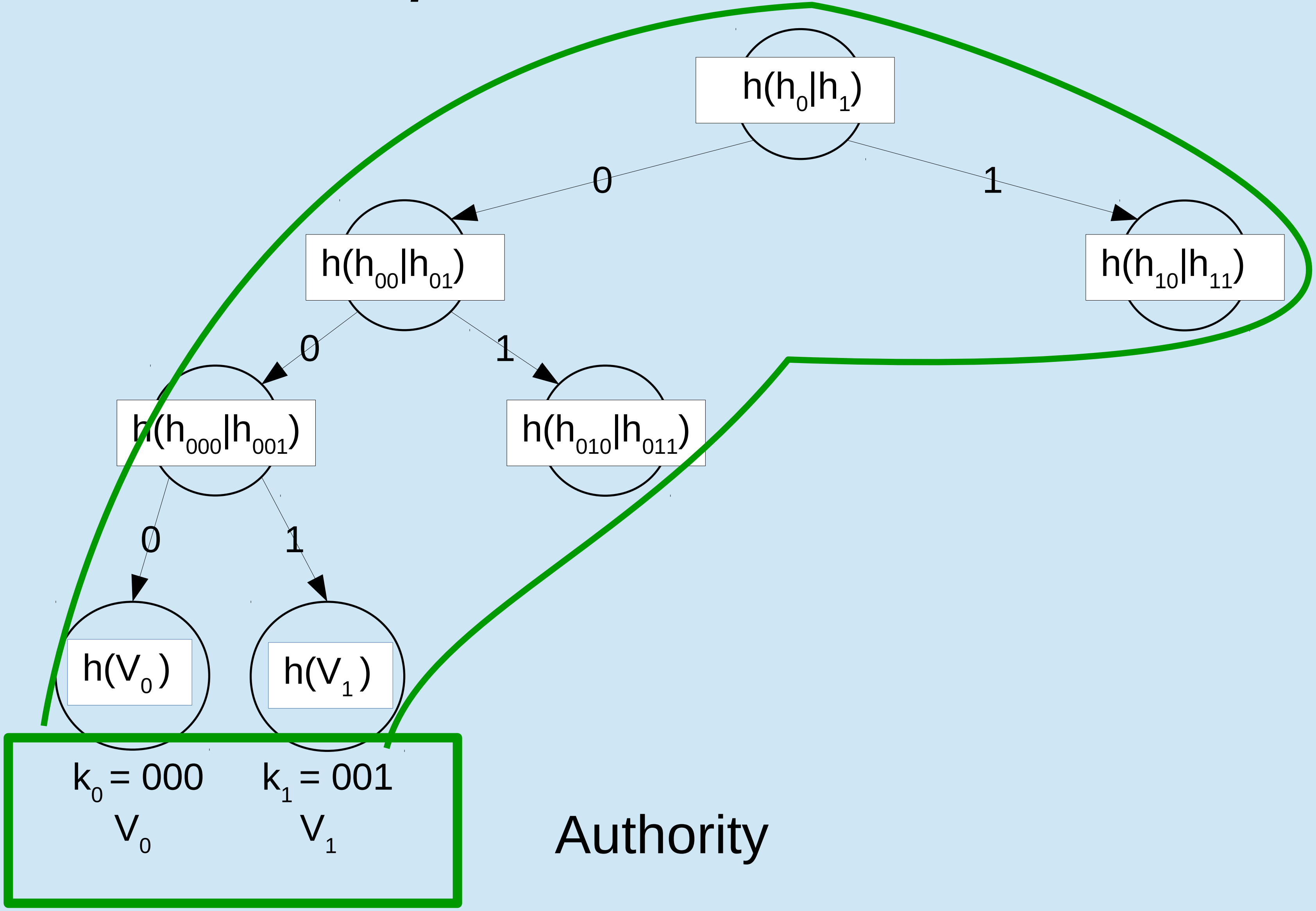
OUR Storage Role: *the small Ledger*

MHT + DHT = ***pruned Authenticated Data Structure*** *(pADS)*



OUR Storage Role: *the small Ledger*

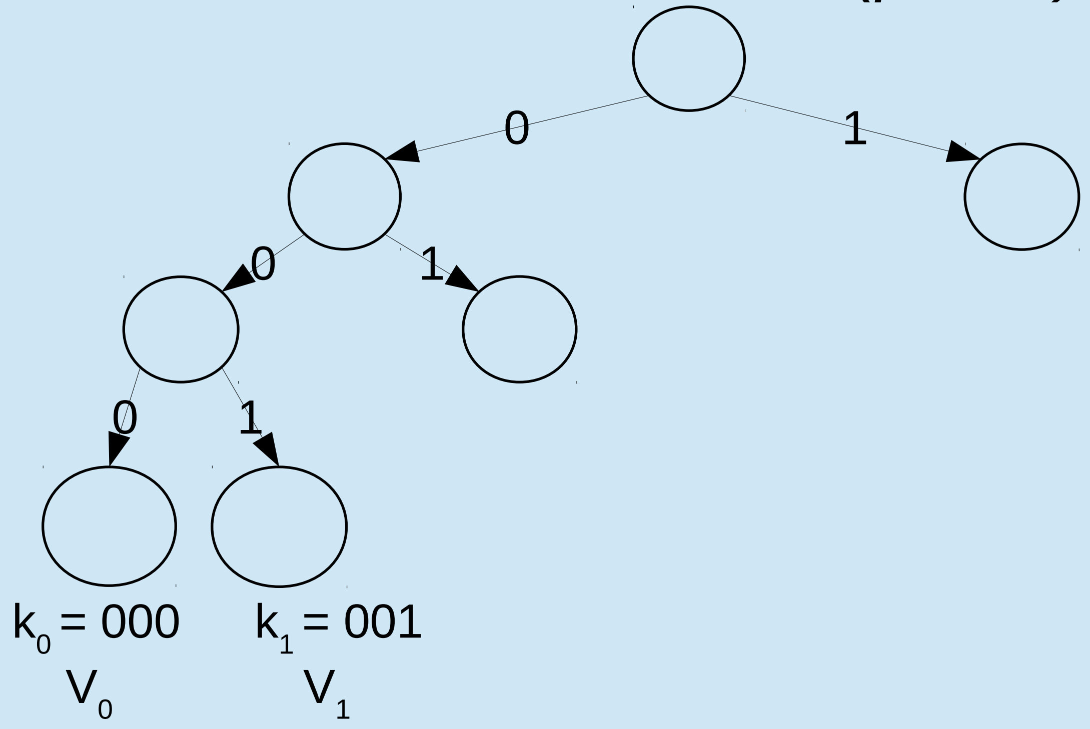
MHT + DHT = ***pruned Authenticated Data Structure*** *(pADS)*



Authority

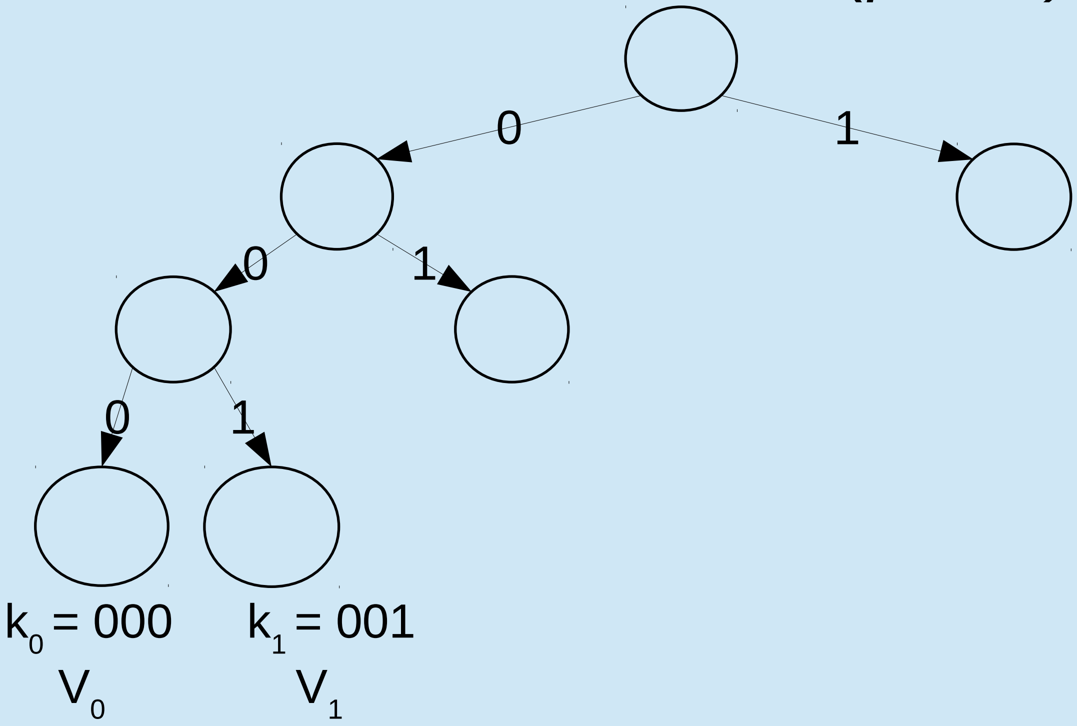
OUR Storage Role: *the small Ledger*

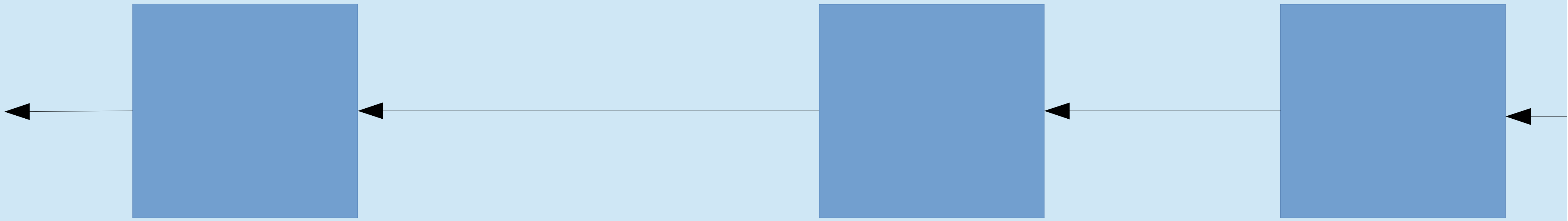
pruned Authenticated Data Structure *(pADS)*



OUR Storage Role: *the small Ledger*

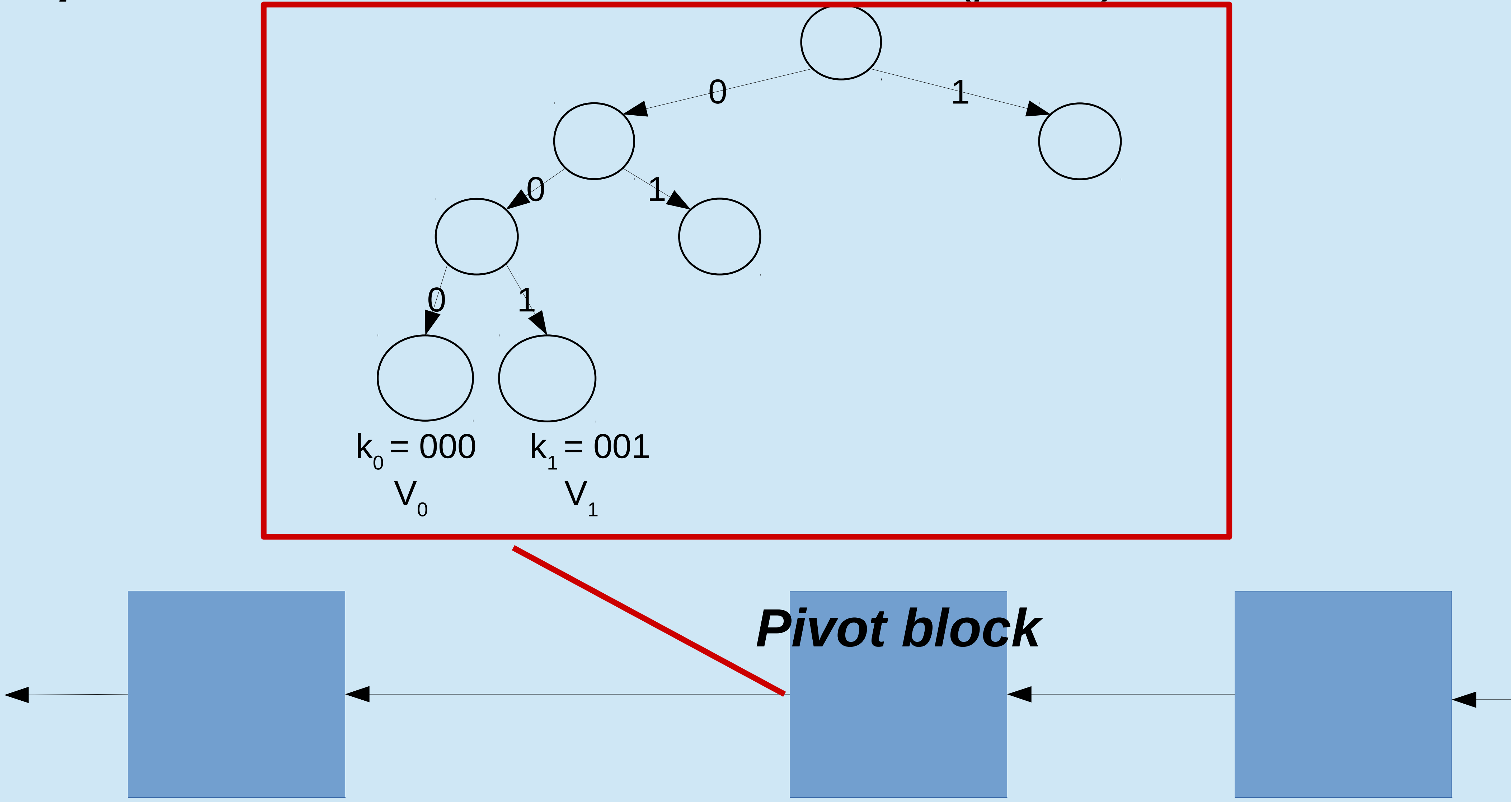
***pruned Authenticated Data Structure** (pADS)*





OUR Storage Role: *the small Ledger*

***pruned Authenticated Data Structure** (pADS)*



OUR Storage Role: ***Transaction***

Traditional Transaction

- *Sender(s)*
- *Receiver(s)*
- *Operation(s)*
- *Signature(s)*

Key 'a':

→ *Value*

→ *Proof*

→ *number of blo*

Key 'b':

→ *Value*

→ *Proof*

→ *number*

Key 'c':

→ *Value*

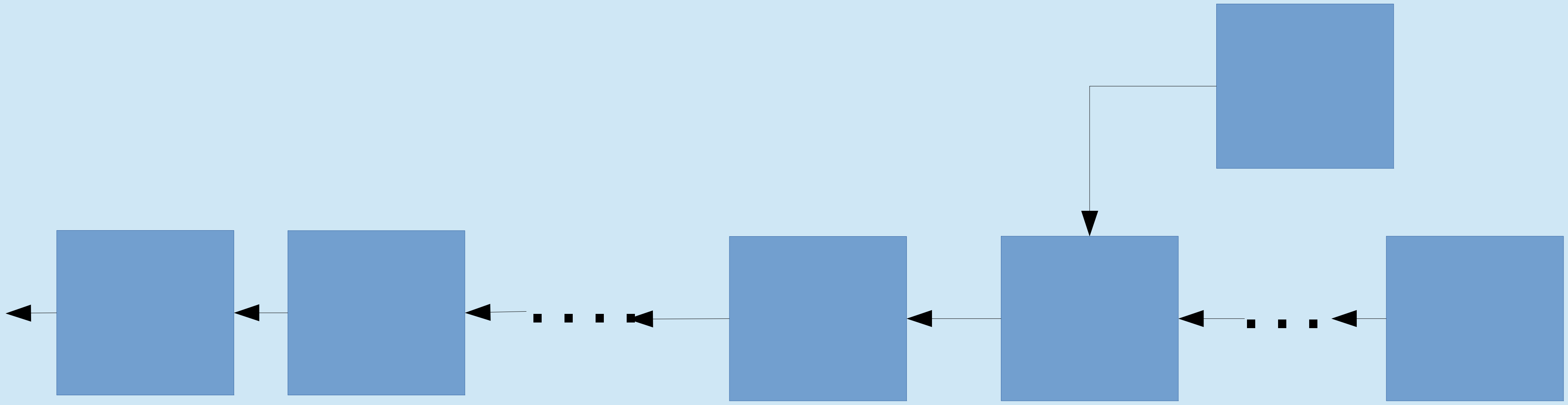
→ *Proof*

→ *n. pivot block*

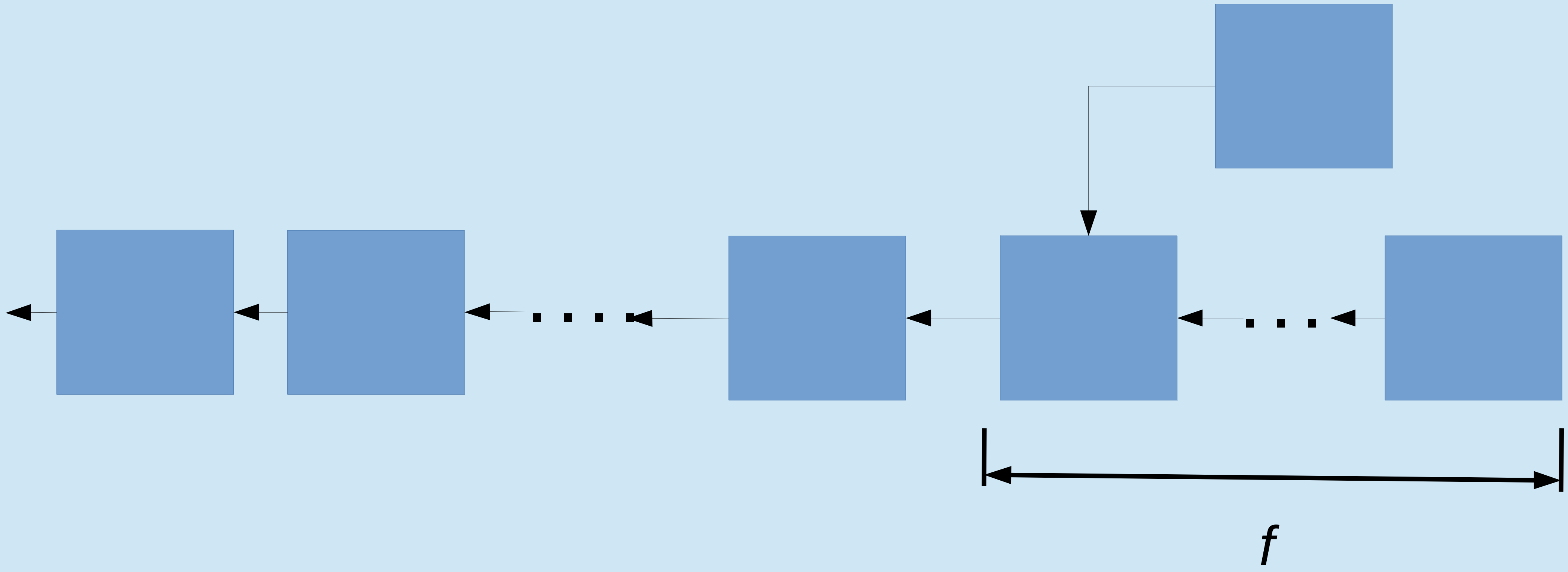
ROMA TRE

Validation Role

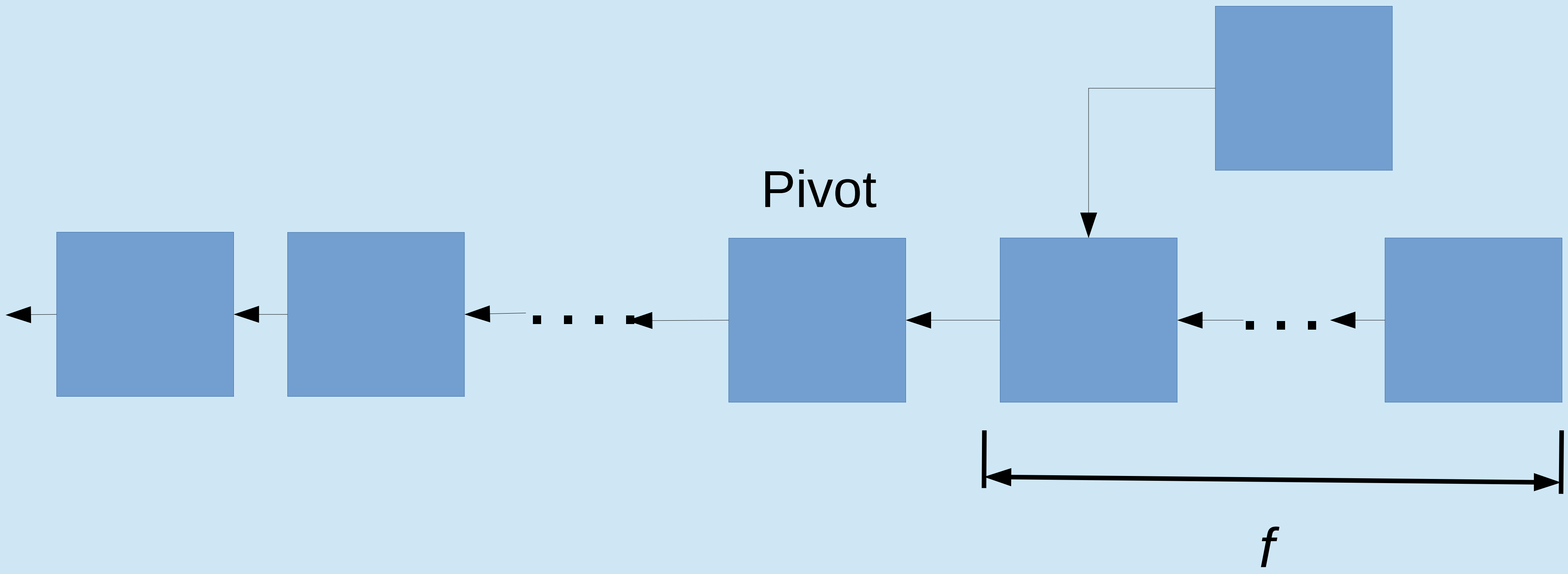
OUR Validation Role: ***truncated*** *blockchain*



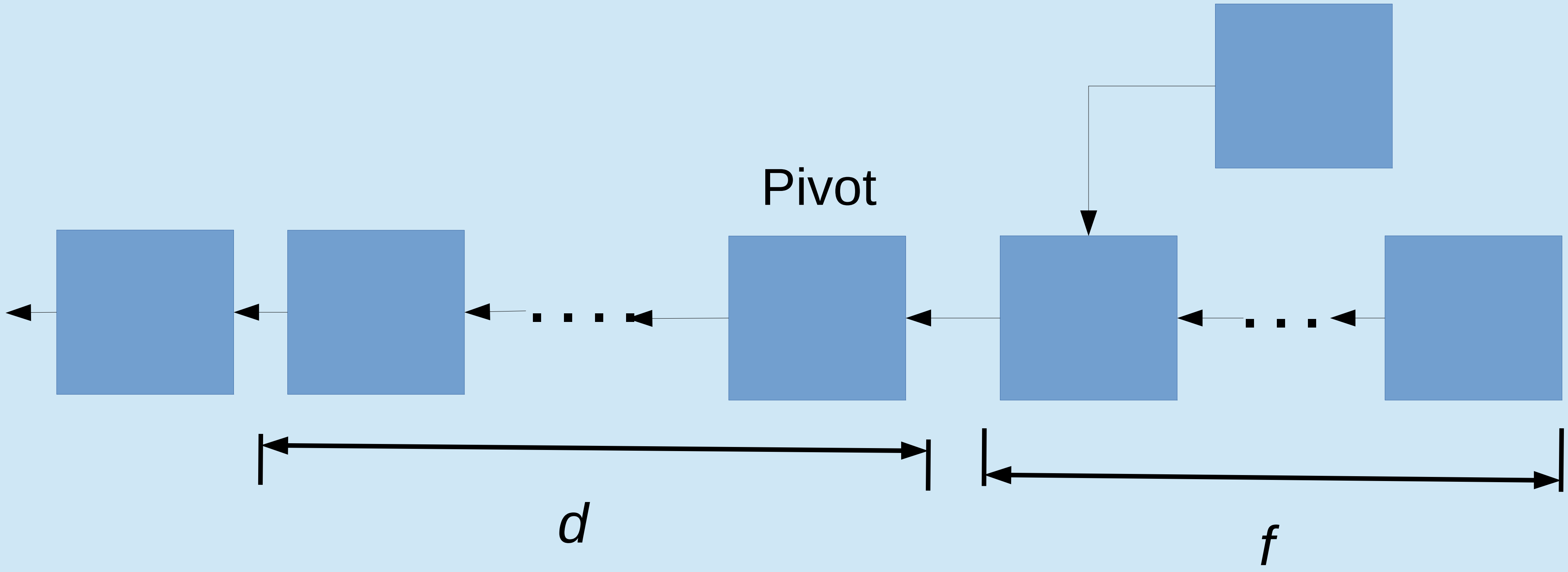
OUR Validation Role: ***truncated*** *blockchain*



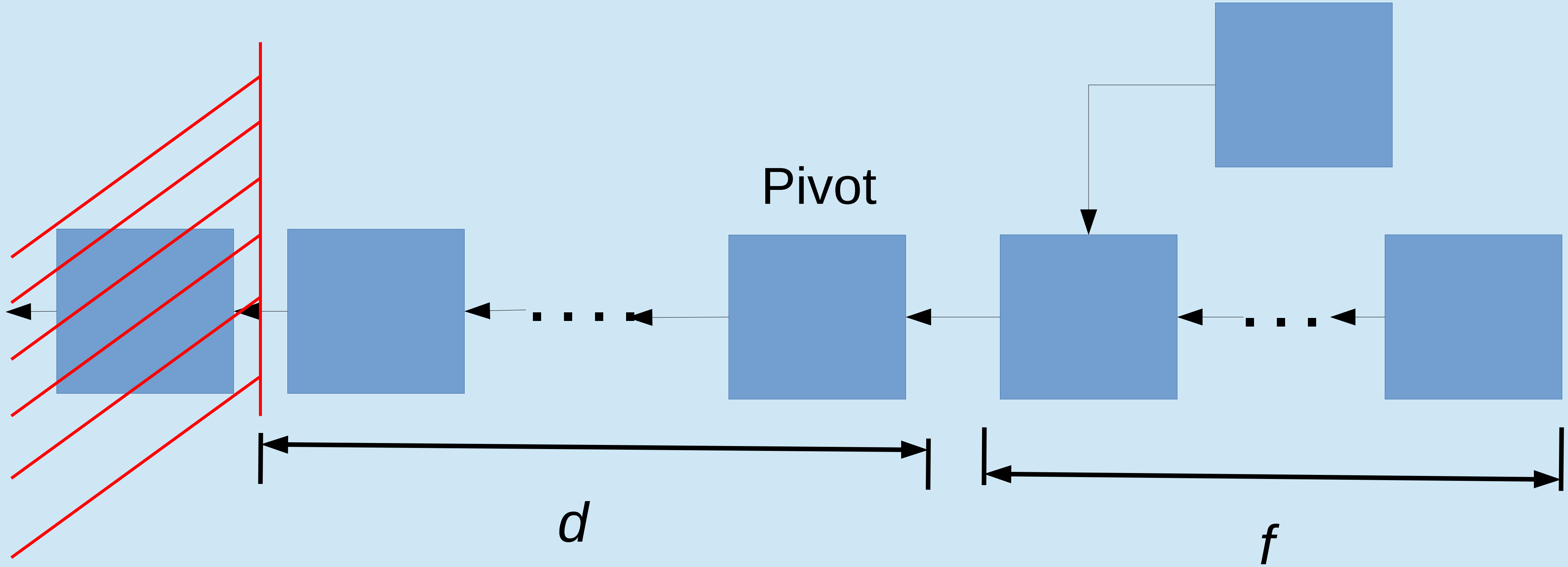
OUR Validation Role: ***truncated** blockchain*



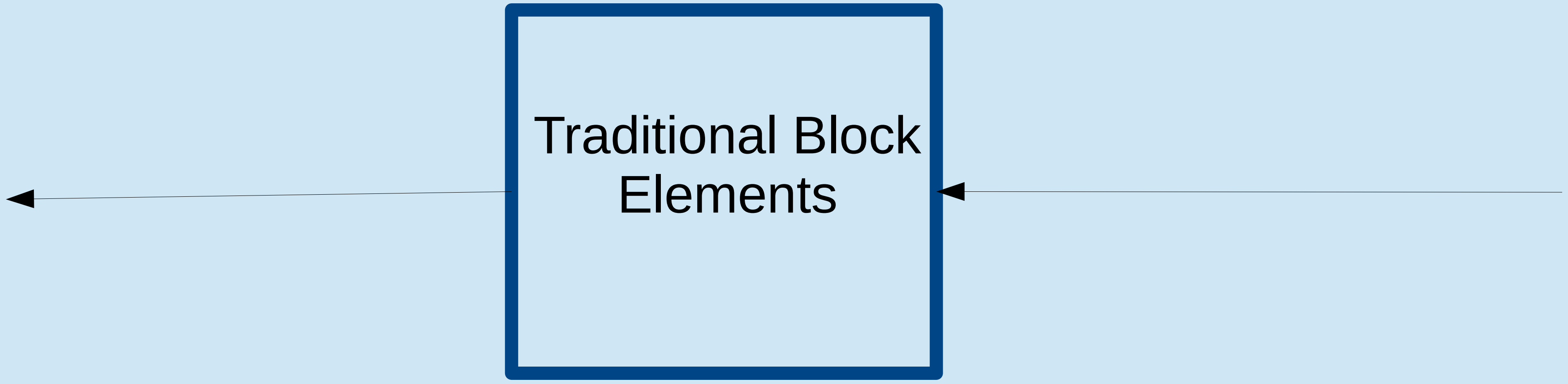
OUR Validation Role: ***truncated*** *blockchain*



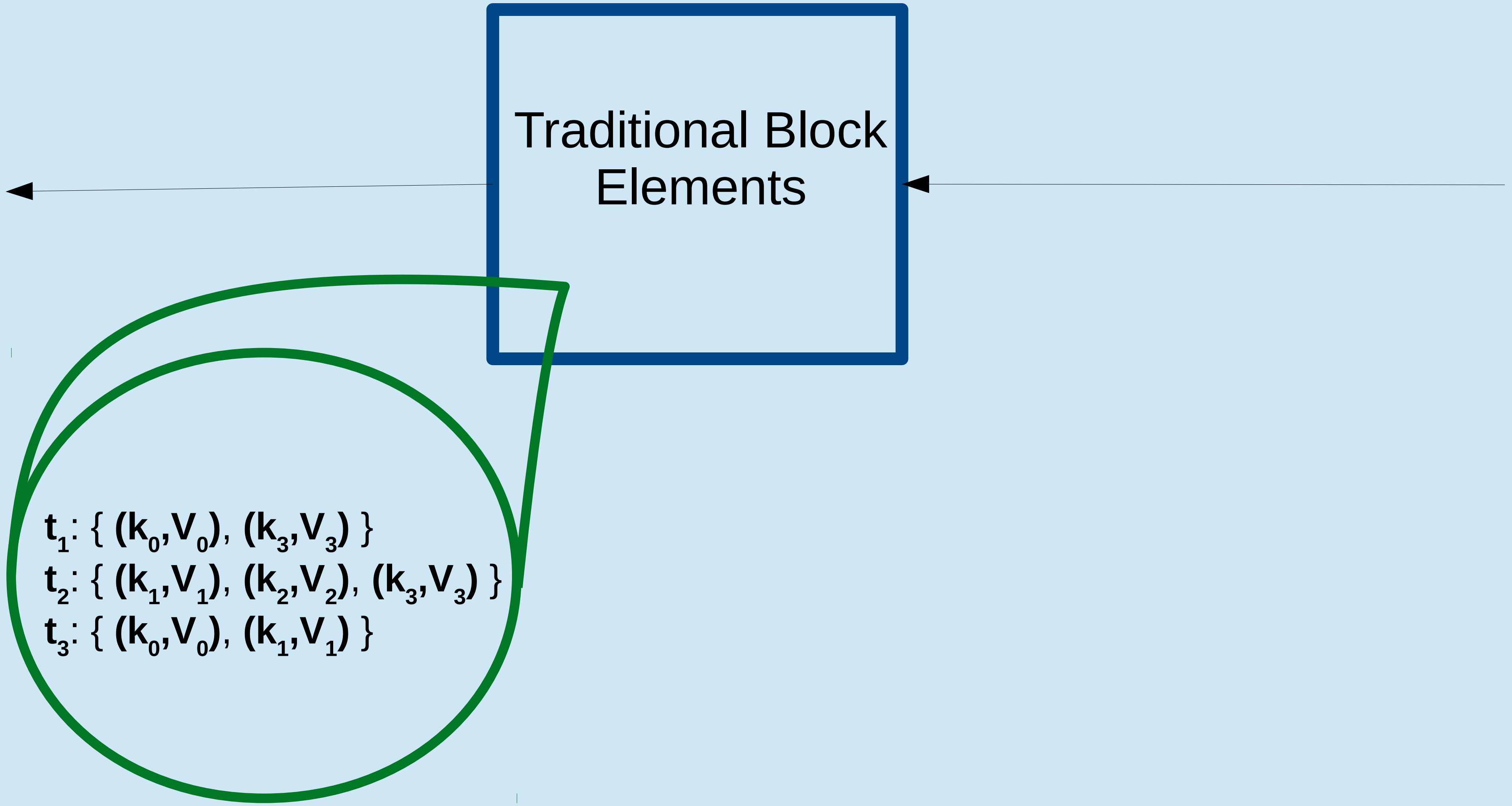
OUR Validation Role: ***truncated** blockchain*



OUR Validation Role: *block schema*



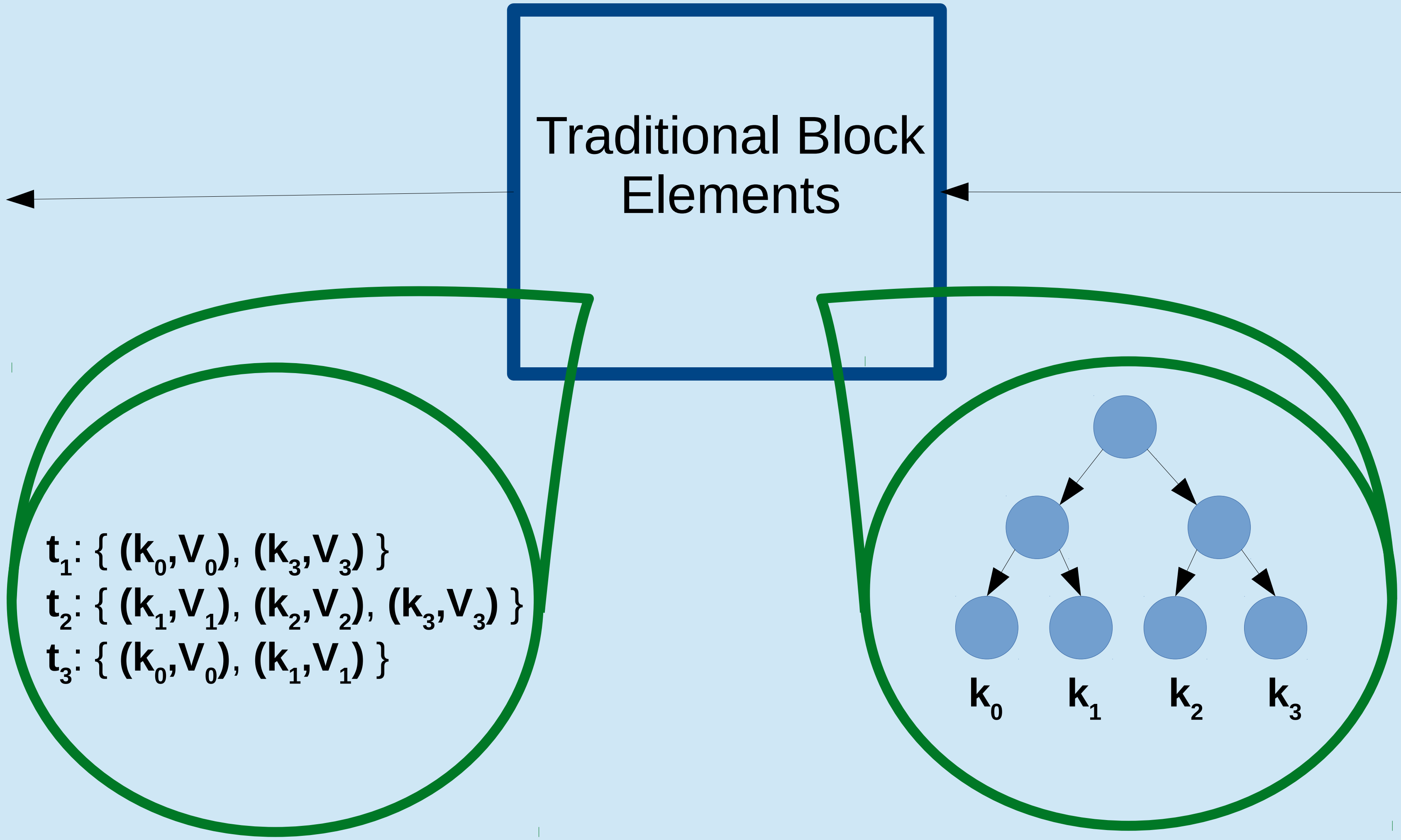
OUR Validation Role: *block schema*



t_1: { (k_0,V_0), (k_3,V_3) }
t_2: { (k_1,V_1), (k_2,V_2), (k_3,V_3) }
t_3: { (k_0,V_0), (k_1,V_1) }

keys involved in transactions with their values

OUR Validation Role: *block schema*



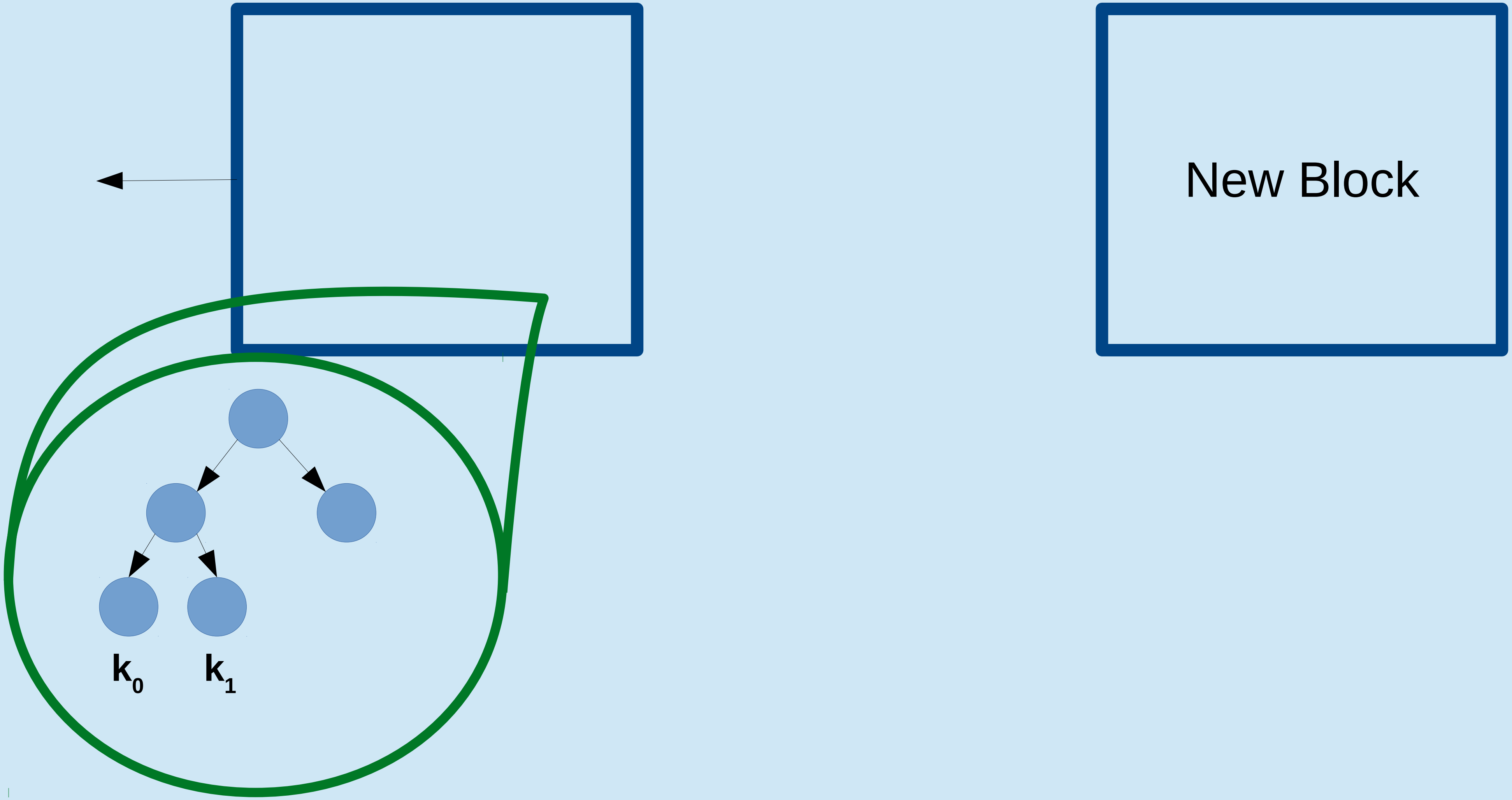
Traditional Block Elements
t_1: { (k_0,V_0), (k_3,V_3) }
t_2: { (k_1,V_1), (k_2,V_2), (k_3,V_3) }
t_3: { (k_0,V_0), (k_1,V_1) }
k_0
k_1
k_2
k_3

keys involved in transactions with their values

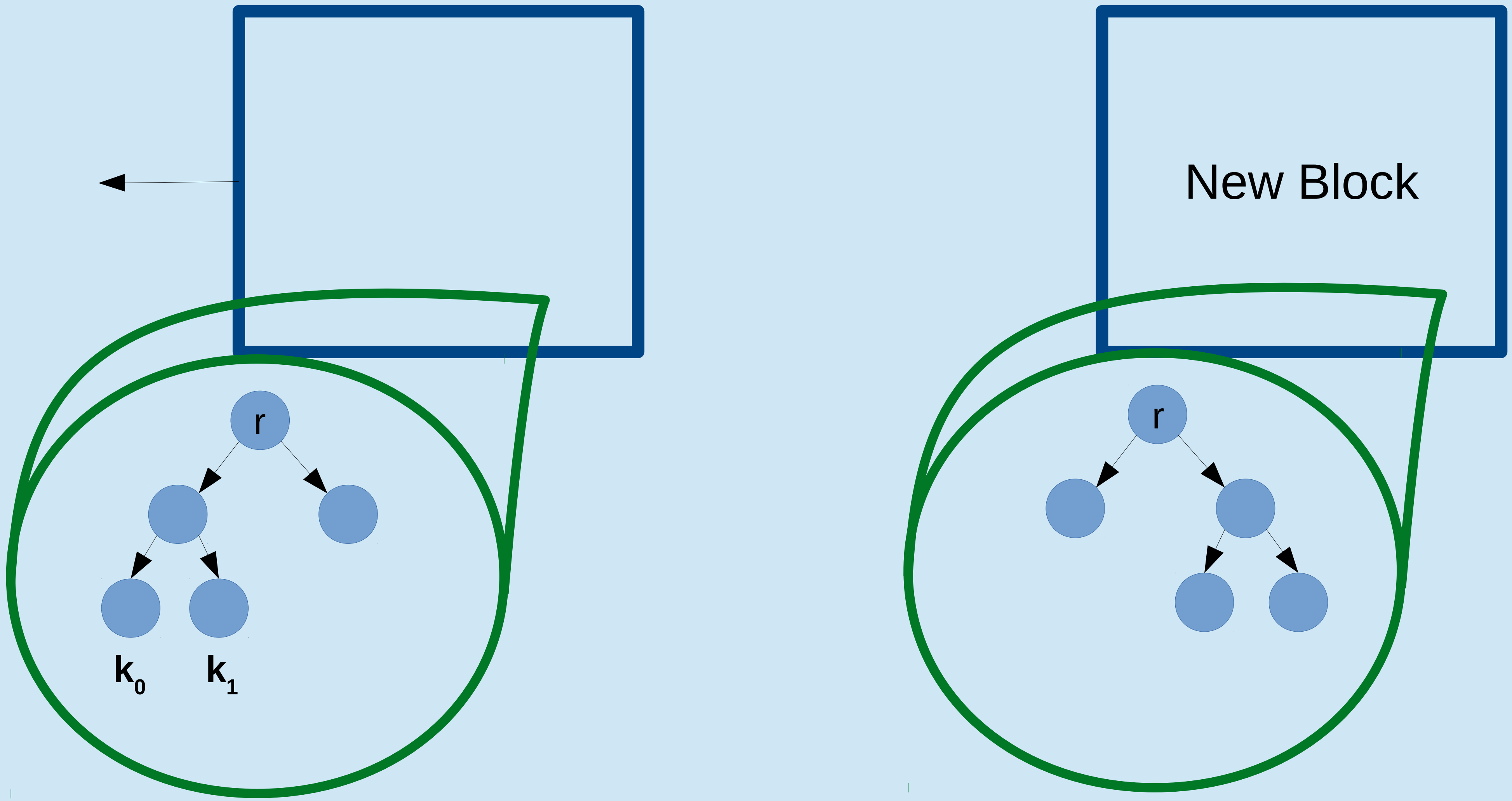
pADS **τ**

OUR Validation Role: **Validation** *block*

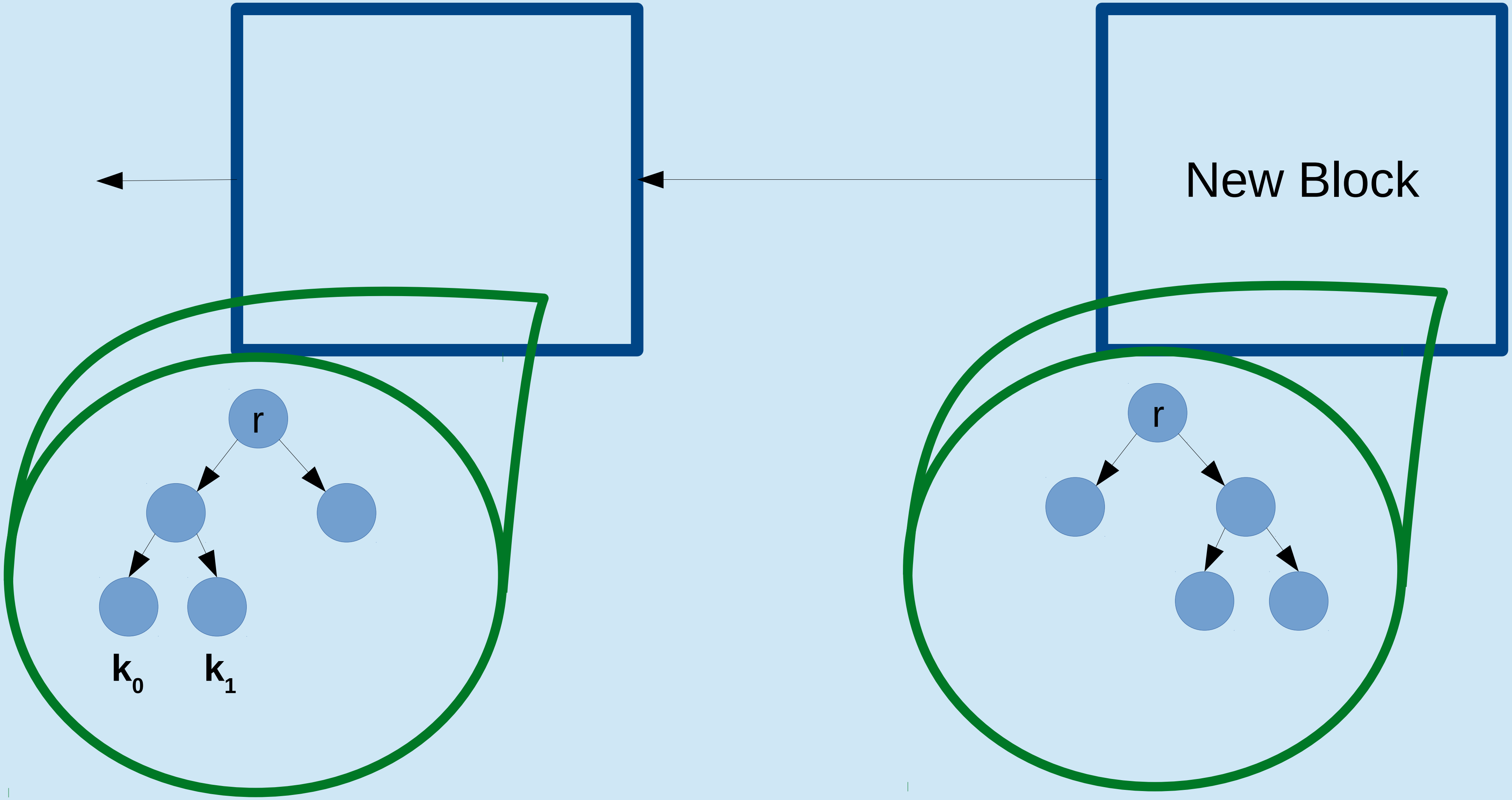
OUR Validation Role: **Validation** *block*



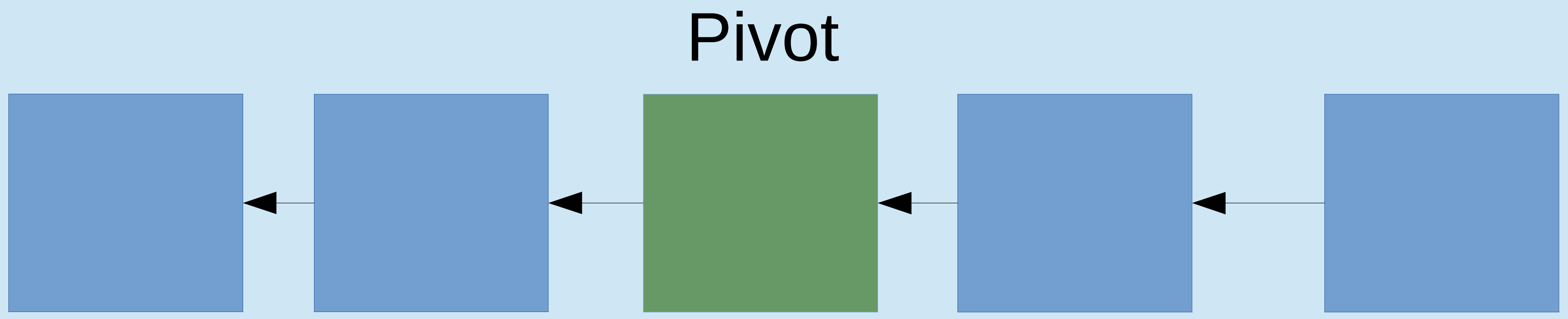
OUR Validation Role: **Validation** *block*



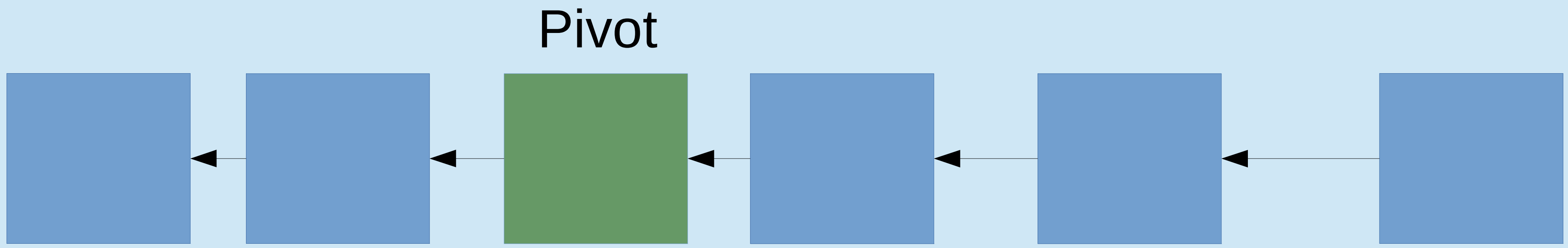
OUR Validation Role: **Validation** *block*



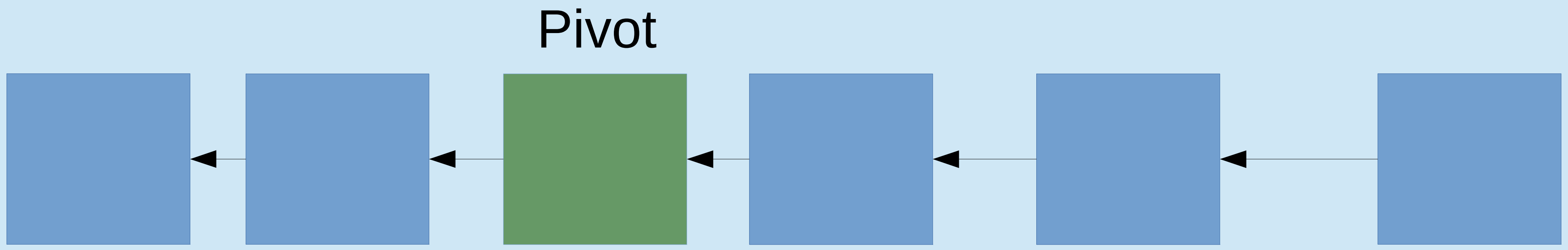
OUR Validation Role: ***Validation*** *block*



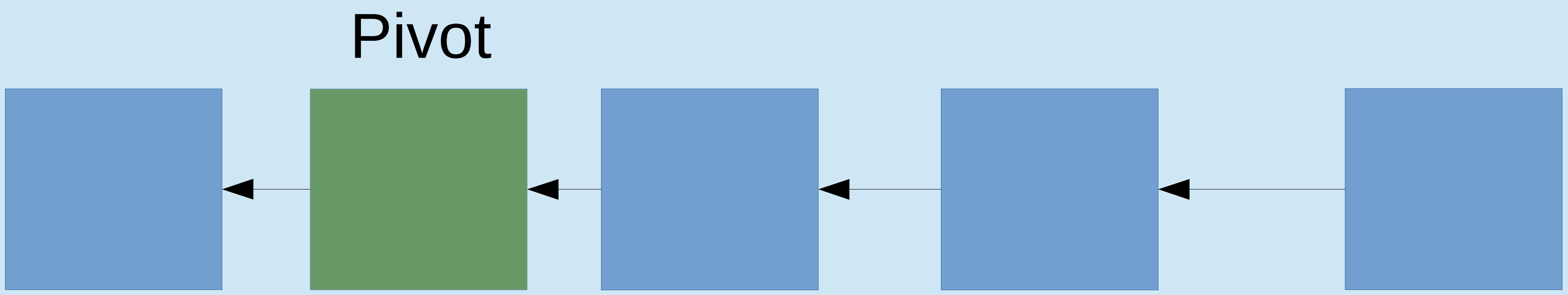
OUR Validation Role: ***Validation*** *block*



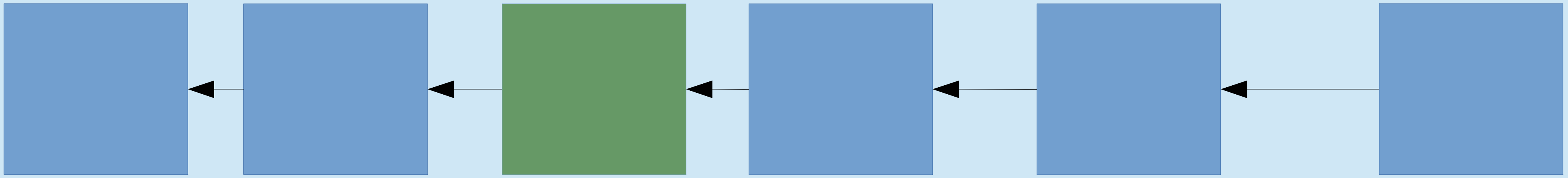
OUR Validation Role: ***Validation*** *block*



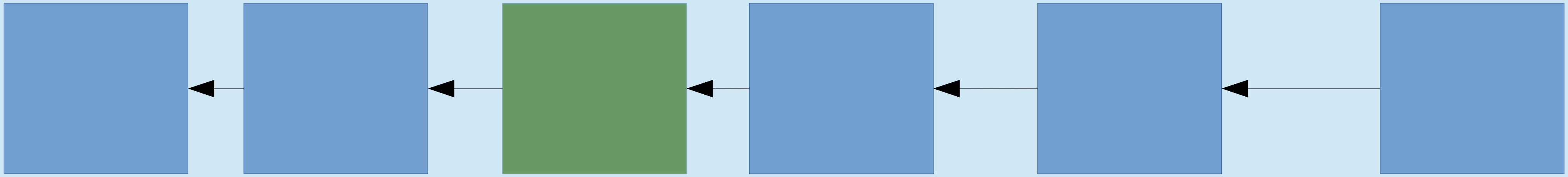
OUR Validation Role: ***Validation** block*



OUR Validation Role: **Mining** *block*



OUR Validation Role: **Mining** *block*

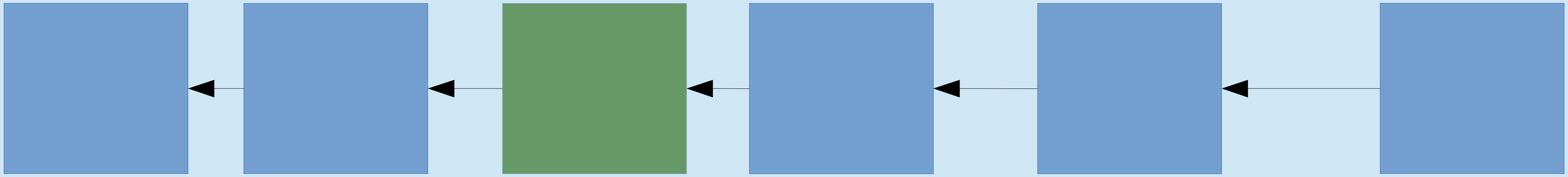


t_1: { (k_2,V_2), (k_3,V_3) , proofs, n. pivot blocks }

t_2: { (k_1,V_1), (k_2,V_2), (k_3,V_3) , proofs, n. pivot blocks }

t_3: { (k_0,V_0), (k_3,V_3) , proofs, n. pivot blocks }

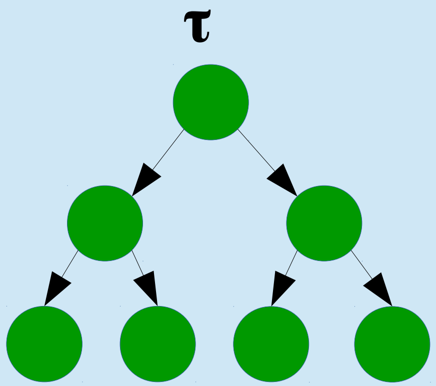
OUR Validation Role: **Mining** *block*



t_1: { (k_2,V_2), (k_3,V_3) , proofs, n. pivot blocks }

t_2: { (k_1,V_1), (k_2,V_2), (k_3,V_3) , proofs, n. pivot blocks }

t_3: { (k_0,V_0), (k_3,V_3) , proofs, n. pivot blocks }



Recap: Transaction/block lifecycle

Recap: Transaction/block lifecycle

1. The transaction creator gets from DHT proofs of the involved keys

Recap: Transaction/block lifecycle

1. The transaction creator gets from DHT proofs of the involved keys
2. The transaction creator broadcast the transaction with proofs

Recap: Transaction/block lifecycle

1. The transaction creator gets from DHT proofs of the involved keys
2. The transaction creator broadcast the transaction with proofs
3. Miners use transaction proofs and $\boldsymbol{\tau}$'s of d+f locally stored blocks to authenticate starting values, execute all transactions of the new block and execute (any) consensus algorithm

Recap: Transaction/block lifecycle

1. The transaction creator gets from DHT proofs of the involved keys
2. The transaction creator broadcast the transaction with proofs
3. Miners use transaction proofs and τ 's of d+f locally stored blocks to authenticate starting values, execute all transactions of the new block and execute (any) consensus algorithm
4. The agreed block is broadcasted to all

Recap: Transaction/block lifecycle

1. The transaction creator gets from DHT proofs of the involved keys
2. The transaction creator broadcast the transaction with proofs
3. Miners use transaction proofs and τ 's of d+f locally stored blocks to authenticate starting values, execute all transactions of the new block and execute (any) consensus algorithm
4. The agreed block is broadcasted to all
5. All nodes validate the new block and update their pivot block

Recap: Transaction/block lifecycle

1. The transaction creator gets from DHT proofs of the involved keys
2. The transaction creator broadcast the transaction with proofs
3. Miners use transaction proofs and τ 's of d+f locally stored blocks to authenticate starting values, execute all transactions of the new block and execute (any) consensus algorithm
4. The agreed block is broadcasted to all
5. All nodes validate the new block and update their pivot block
6. DHT nodes that are authority for keys involved in the new block also update their pADS

Conclusion and Future works

- Store a small amount of data **(pADS, DHT, truncated chain)**
- Validation without Ledger **(proof and pivot)**
- First synchronization of a node performed in less than a minute

➔ implementation of this approach
➔ extensive tests

ROMA TRE

Thank you
for your attention