RGB Tech Internals, part III

RGB smart contract data & state details

LNP/BP Standards Association
Prepared & supervised by **Dr Maxim Orlovsky**, **Pandora Core AG**
Created with support from **Bitfinex & Fulgur Ventures**

RGB Tech Internals Parts

I. ~~General overview & introduction to RGB~~

II. ~~Data structures hierarchy~~

III. RGB smart contract data & state details *<- you are here*

IV. Commitments & encoding details
(more on single-use-seals, deterministic bitcoin commitments, multi-message commitments, strict encoding etc)

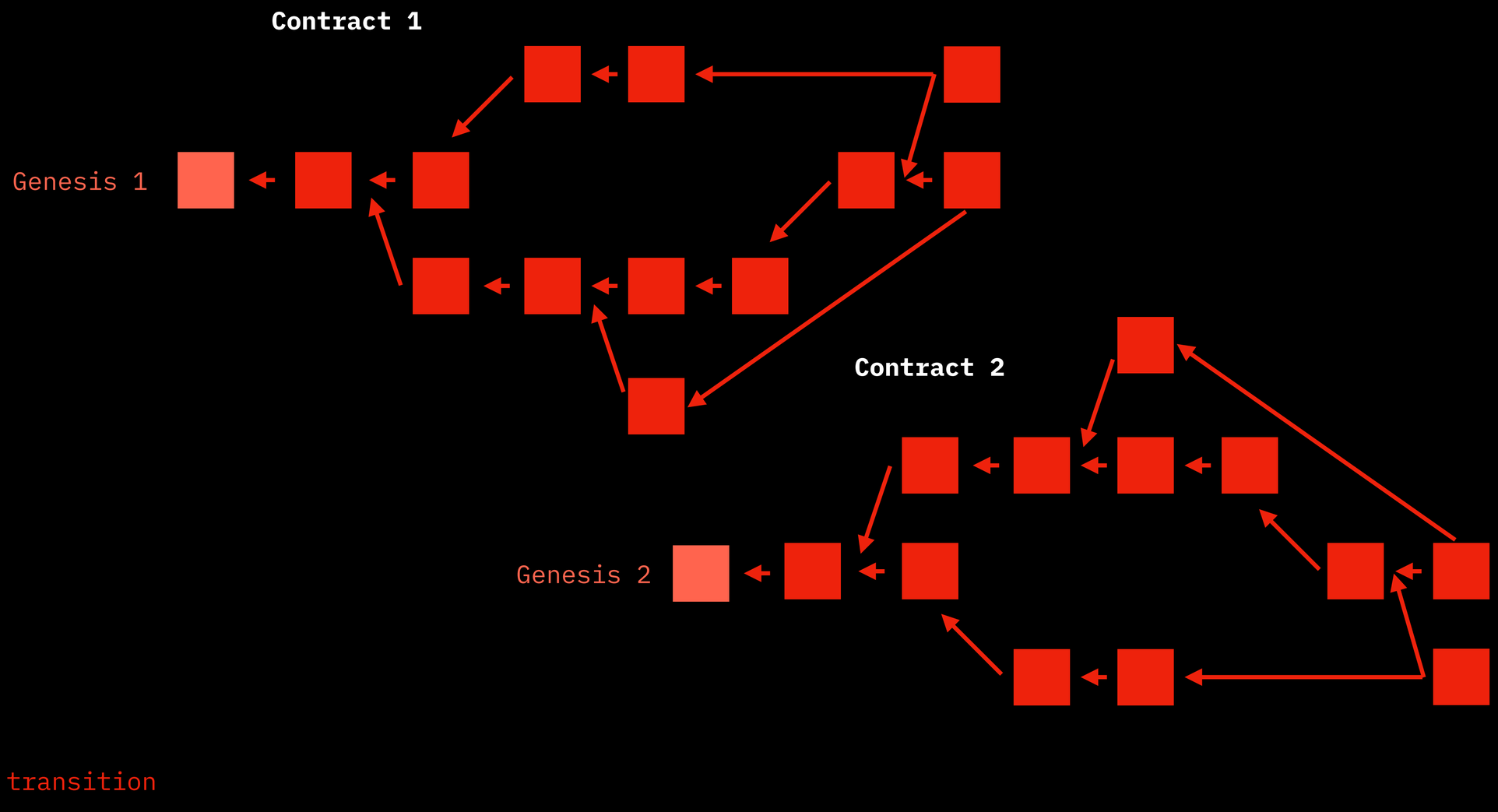
V. Schema & scripting. Validation.

VI. Networking: protocols and infrastructure

VII. Lightning network specifics

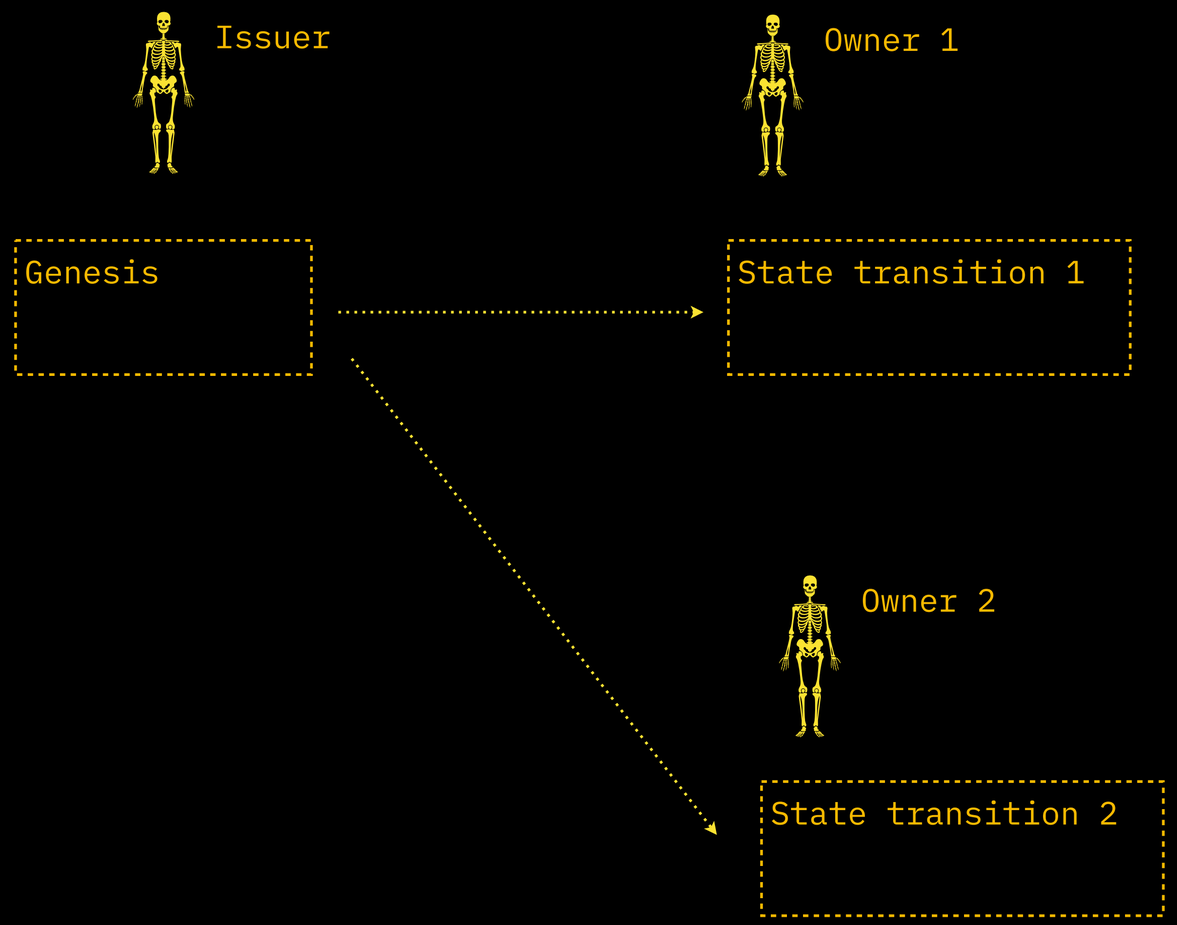
VIII. Ready for the future
(Schnorr's & Taproot, eltoo, bi-directional channels, DLCs, channel factories, channel splits, multi-peer channels)

RGB client-side view of multiple contract DAGs



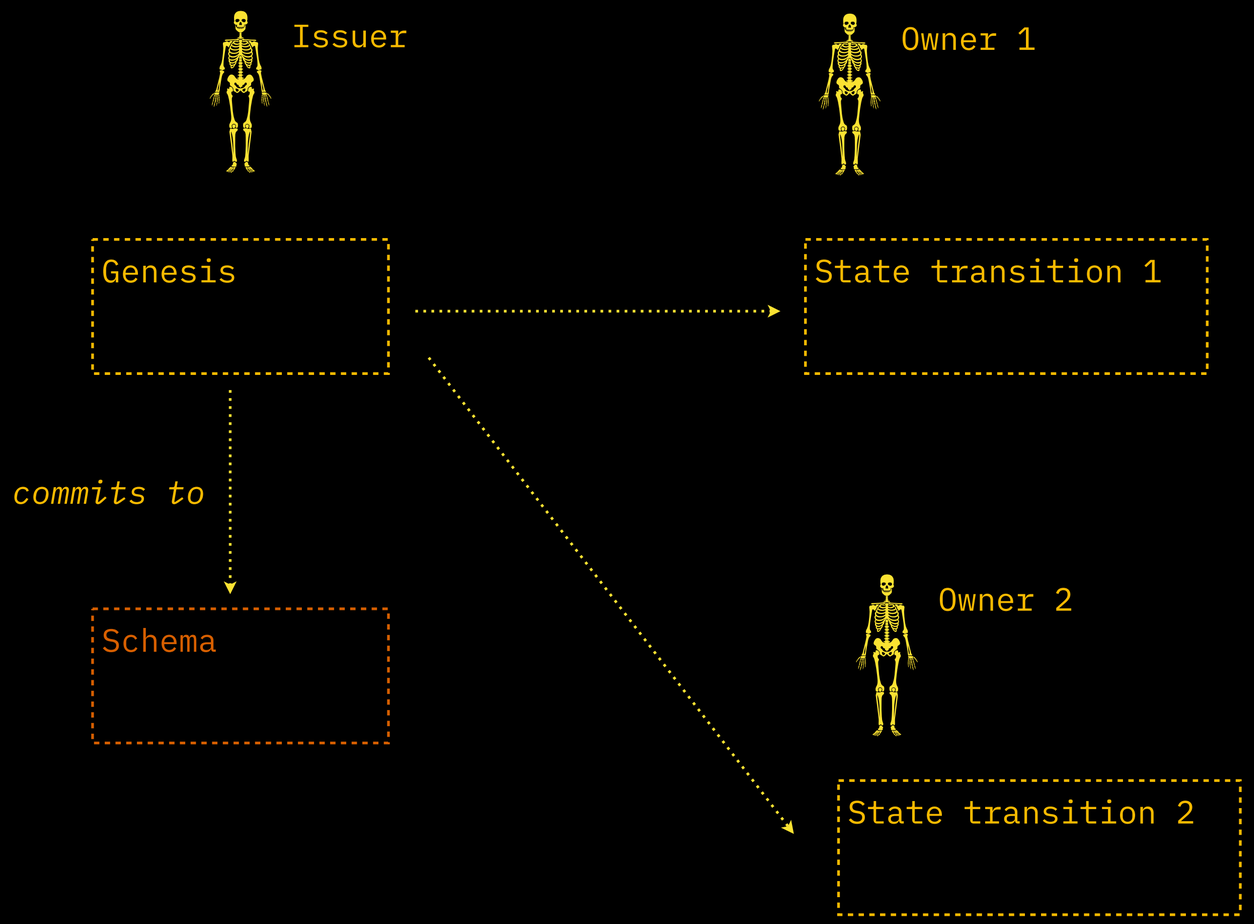
From components to data structures

- Genesis
- State transition(s)
- DAG of state transitions



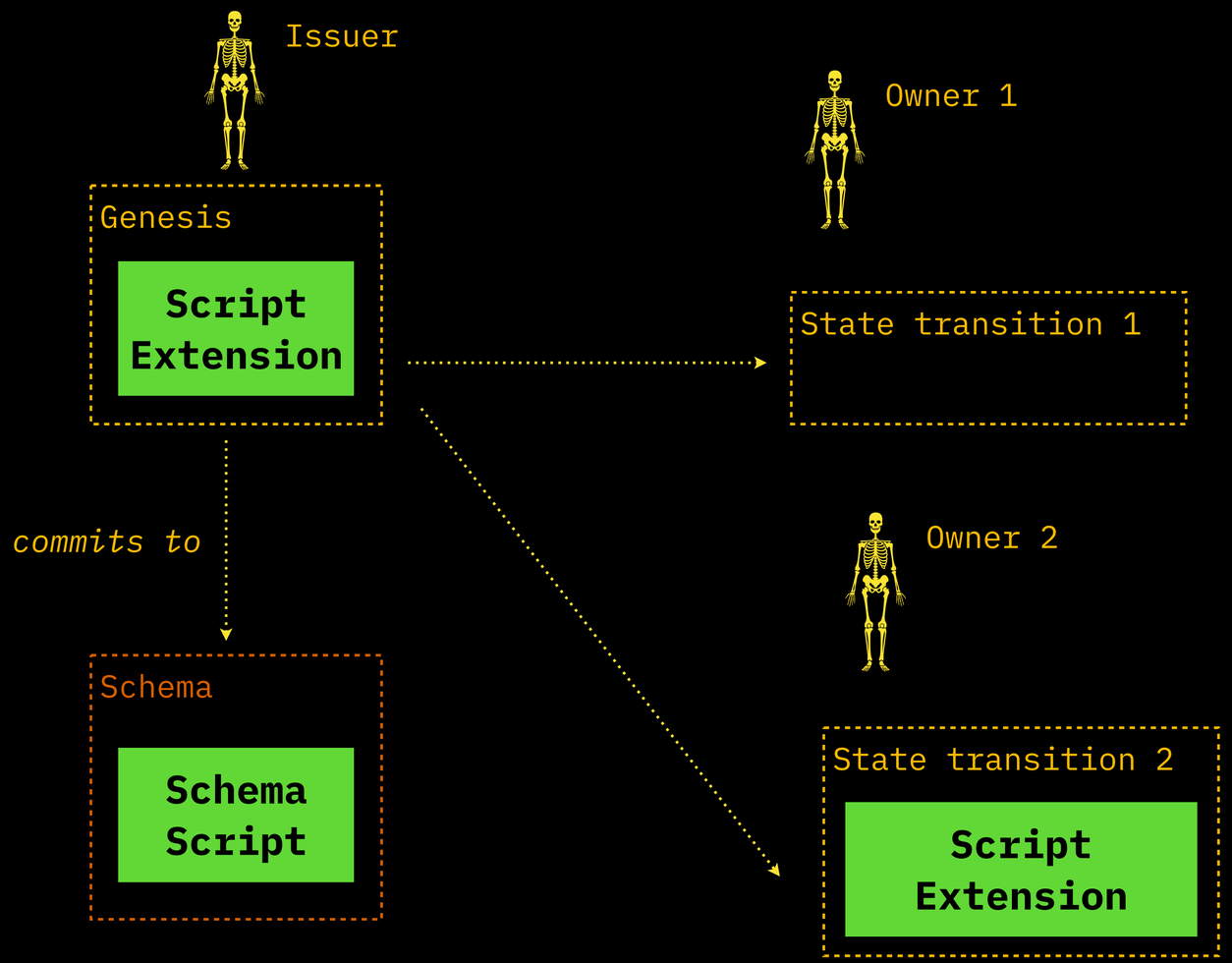
From components to data structures

- Genesis
- State transition(s)
- DAG of state transitions
- Schema (used by multiple geneses)



From components to data structures

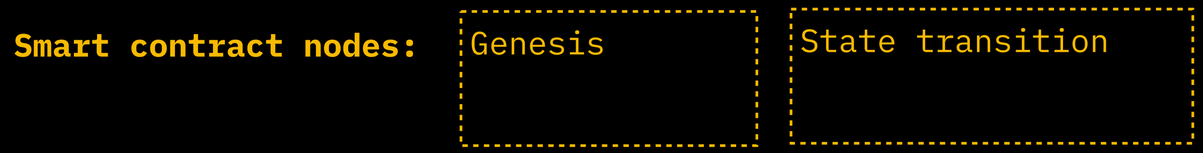
- Genesis
- State transition(s)
- DAG of state transitions
- Schema (used by multiple geneses)
- Scripts
  - Schema-level scripts
  - Genesis-level scripts (optional, within schema-defined restrictions)
  - Scripts extensions (optional, within schema-& genesis-defined restrictions)



From components to data structures

- Smart contract nodes
  - Genesis:
    - one per contract
    - uniquely identifies the contract
    - created by issuer
  - State transition
    - DAG, i.e. many per contract
    - created by previous owner to transfer some state to a new owner

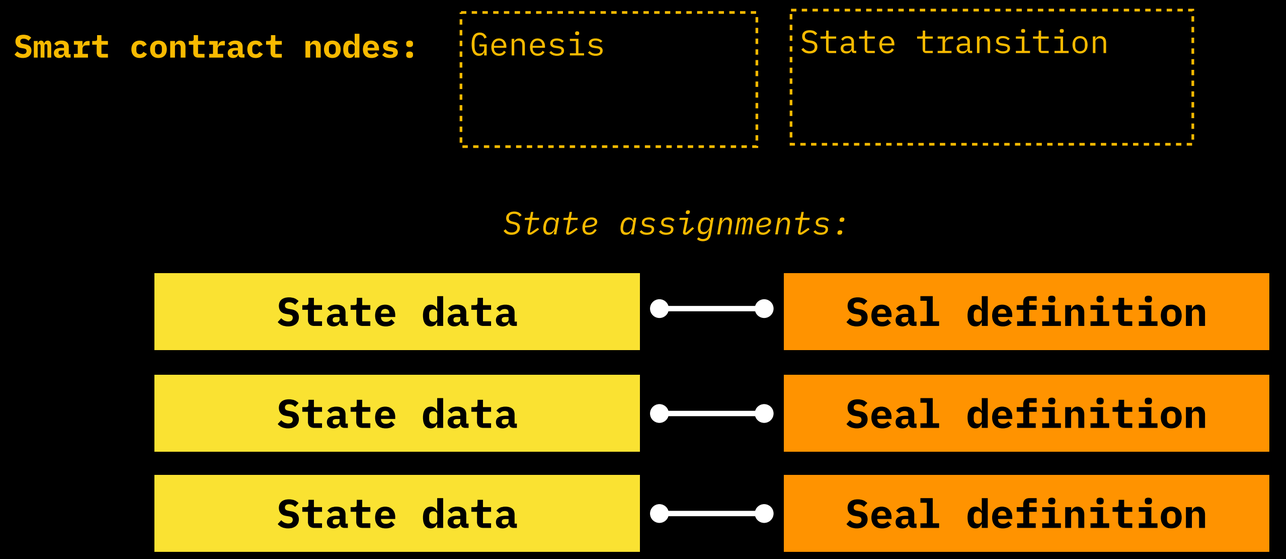
Smart contract nodes assign some state to seals; i.e. define ownership structure



From components to data structures

- Smart contract nodes
  - Genesis:
    - one per contract
    - uniquely identifies the contract
    - created by issuer
  - State transition
    - DAG, i.e. many per contract
    - created by previous owner to transfer some state to a new owner

Smart contract nodes assign some state to seals; i.e. define ownership structure



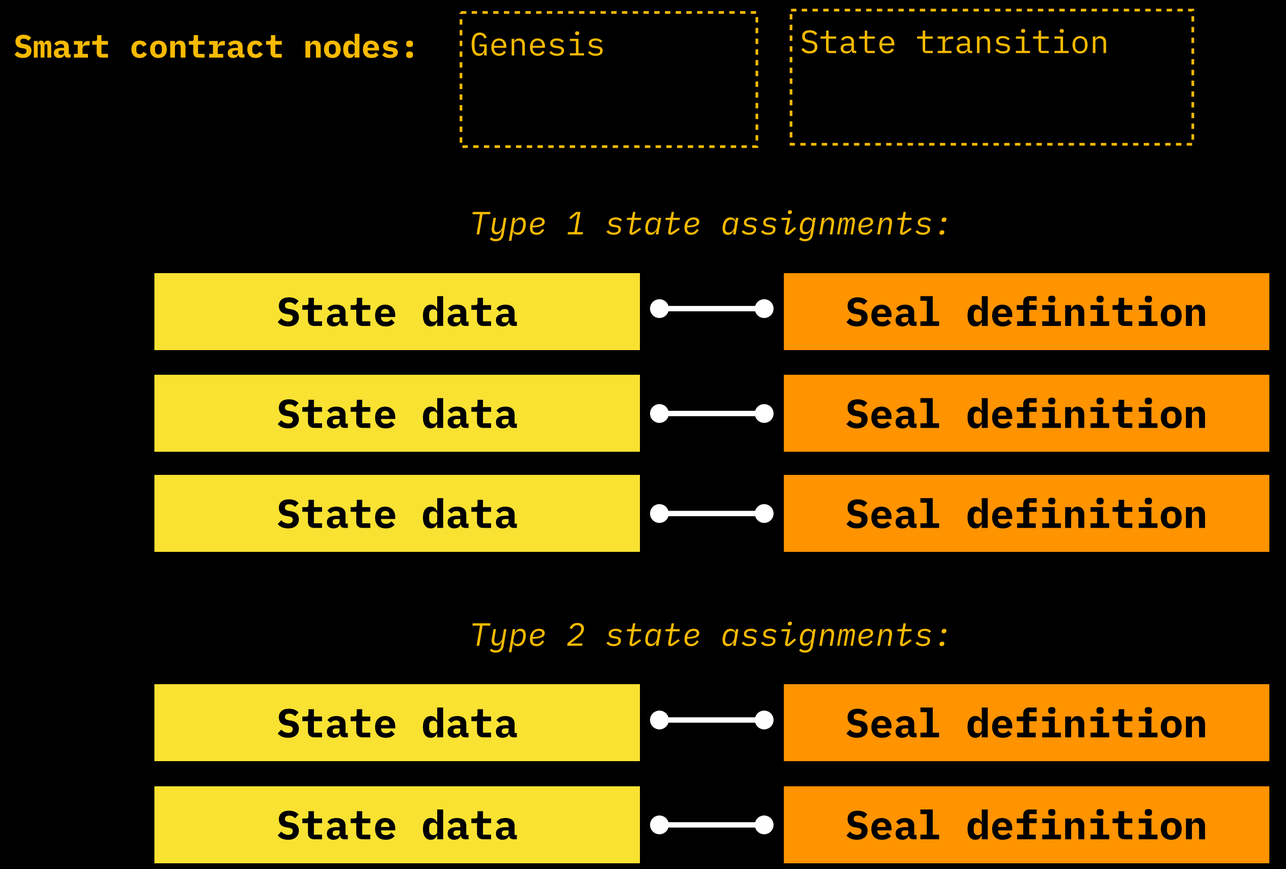
Data, State, Assignments

- **State** is any data of a well-defined type (named **data type**)
  - The same data types are used in metadata, except for homomorphically-encrypted
- **Assignment type** — semantic class for a given digital right with defined data type
- **Assignment** = State data + seal definition
  - State data & seal definition can be independently **concealed**
- **Assignment variant** – a set of assignments of certain type

- State assignment = assignment of some rights
  - Assignment type defines class of the rights
  - State defines parameters of those rights
  - Seal definition defines ownership
- There can be multiple assignments under some specific assignment type (like multiple token allocations to different transaction outputs)

From components to data structures

- Each smart contract node may define many types of state assignments, as far as it is allowed by the schema



State type system: isolation paradigm

- **State** can be of different **assignment type**
- States of different types are isolated from each other during validation: both at Simplicity-script and Schema level
- State types are defined by Schema
- Assignment type != state **data format**:
  the later defines computer data format (integer, string etc)
  the former defines semantics (meaning in terms of smart contract)

Example: contracts under assets schema

- Assignment types:
  - Asset ownership right: homomorphically-encrypted data
  - Secondary issuance right: “void” data
  - …
- Each state transition may assign asset ownership to many transaction outputs (like one for new owner, one for change)

State & metadata data types

State only

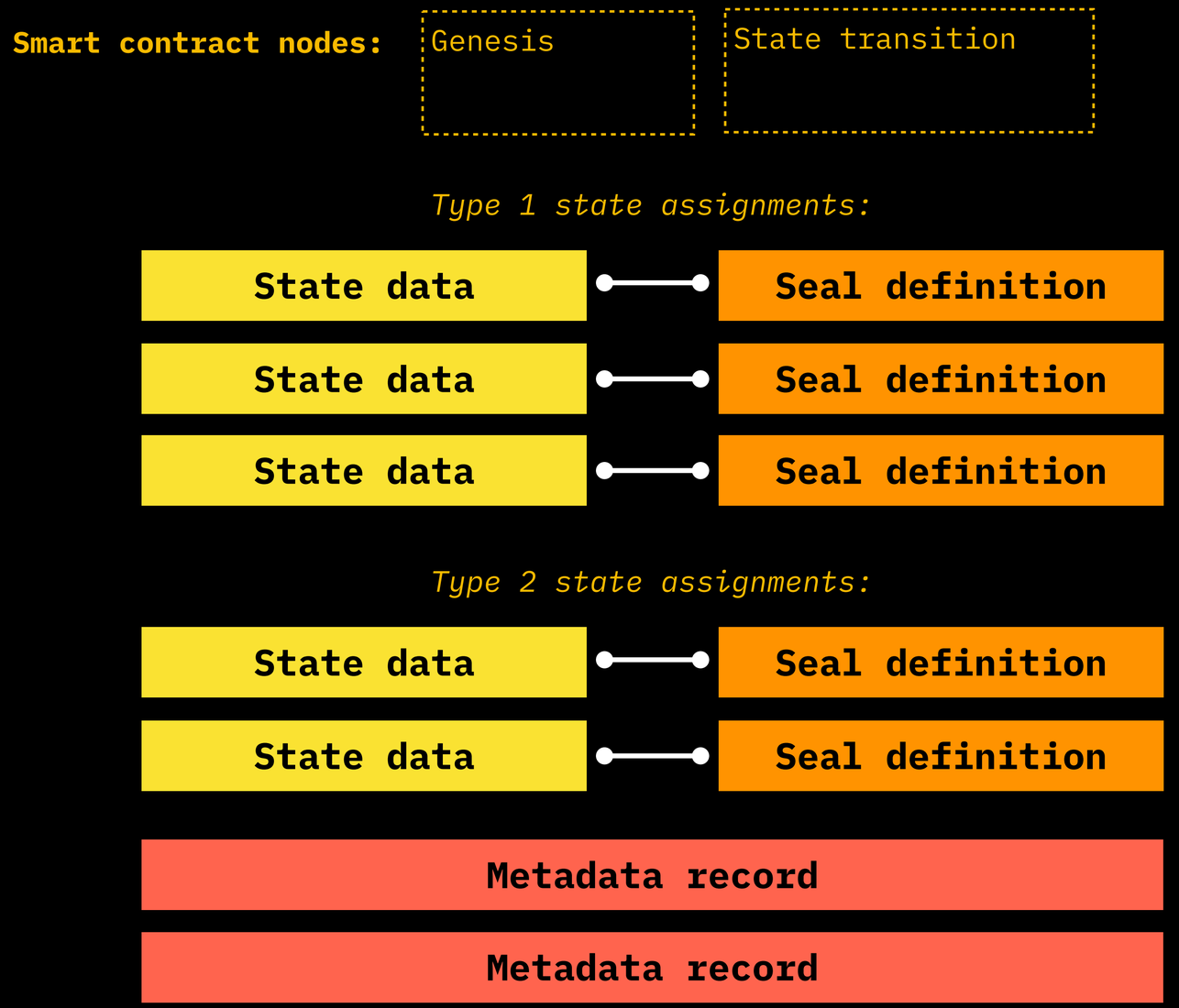
- Homomorphically-encryptable data:
  64-bit values, encrypted with Pedersen commitments (“amount”)
- “Void” data: no data

State & metadata

- Hashable data:
  - Signed & unsigned of different bit dimension
    (from 8- to 256-bit)
  - Float numbers (32- and 64-bit)
  - Strings and byte strings (blobs)
  - Public keys, signatures…

From components to data structures

- Each smart contract node may define many types of state assignments, as far as it is allowed by the schema
- Metadata records have no isolation at both Schema validation rules and Simplicity script levels



State data

- Evolve
- Always assigned to a transaction output (i.e. digital right)
- Validated in isolated mode
- Can contain homomorphically-encrypted data
- Can be concealed, either homomorphically or by hashing, additionally to merklization

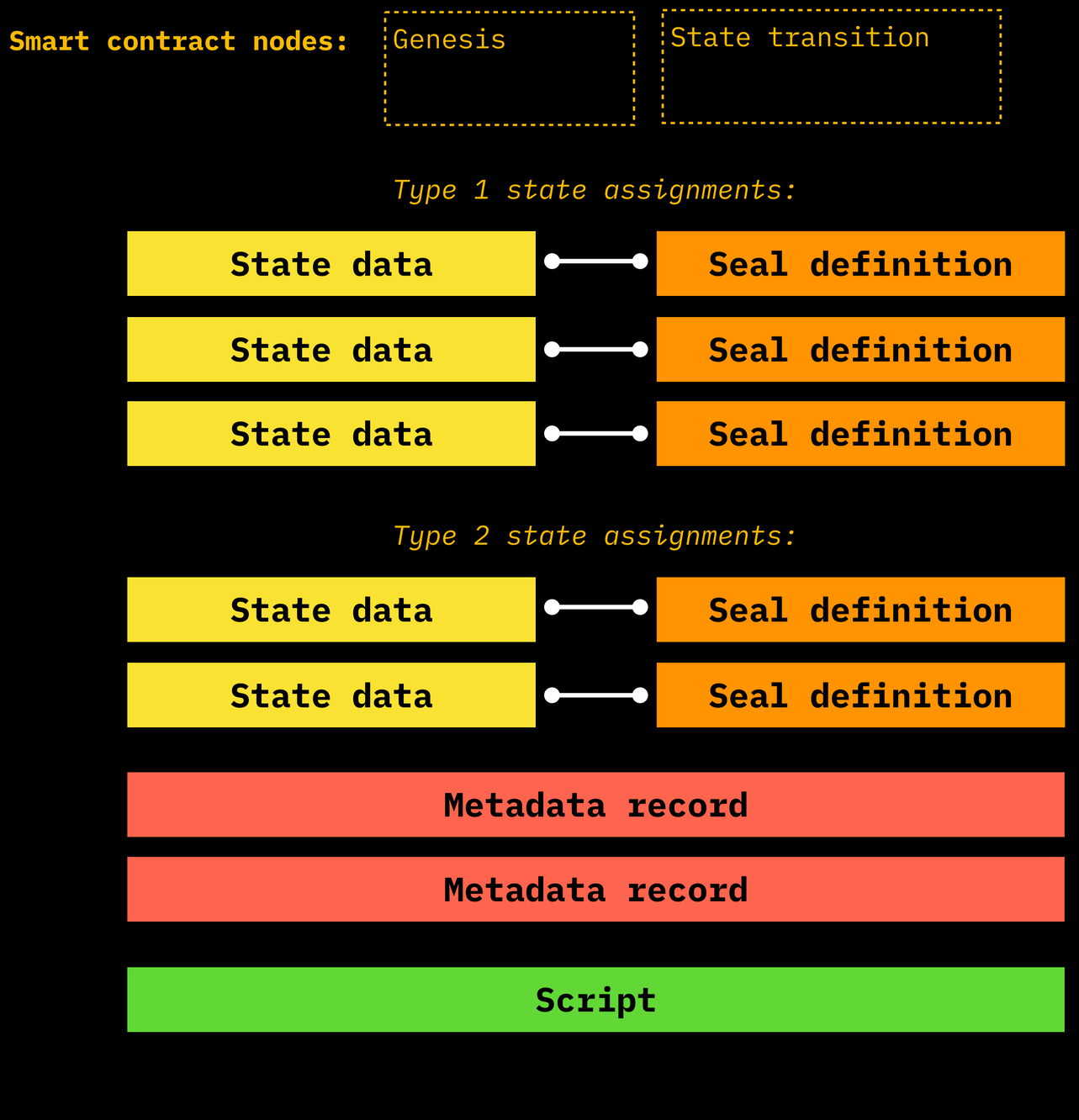
Metadata

- Can be valid for the whole contract scope (like asset name in Genesis)
- Validated in general + provided during validation of each state type (i.e. a way to coordinate validation of different state types)
- Concealed only by merklization and partial reveal

- Share the same non-homomorphic data types
- Share the same structure of occurrences
- Merklized inside smart contract nodes

From components to data structures

- State types
  - State assignment
    - Seal definition
    - State data of specific type
- Metadata field
  - Metadata record
    - State data of specific type
- Simplicity scripts for additional validation of all future transitions starting from this one



From components to data structures

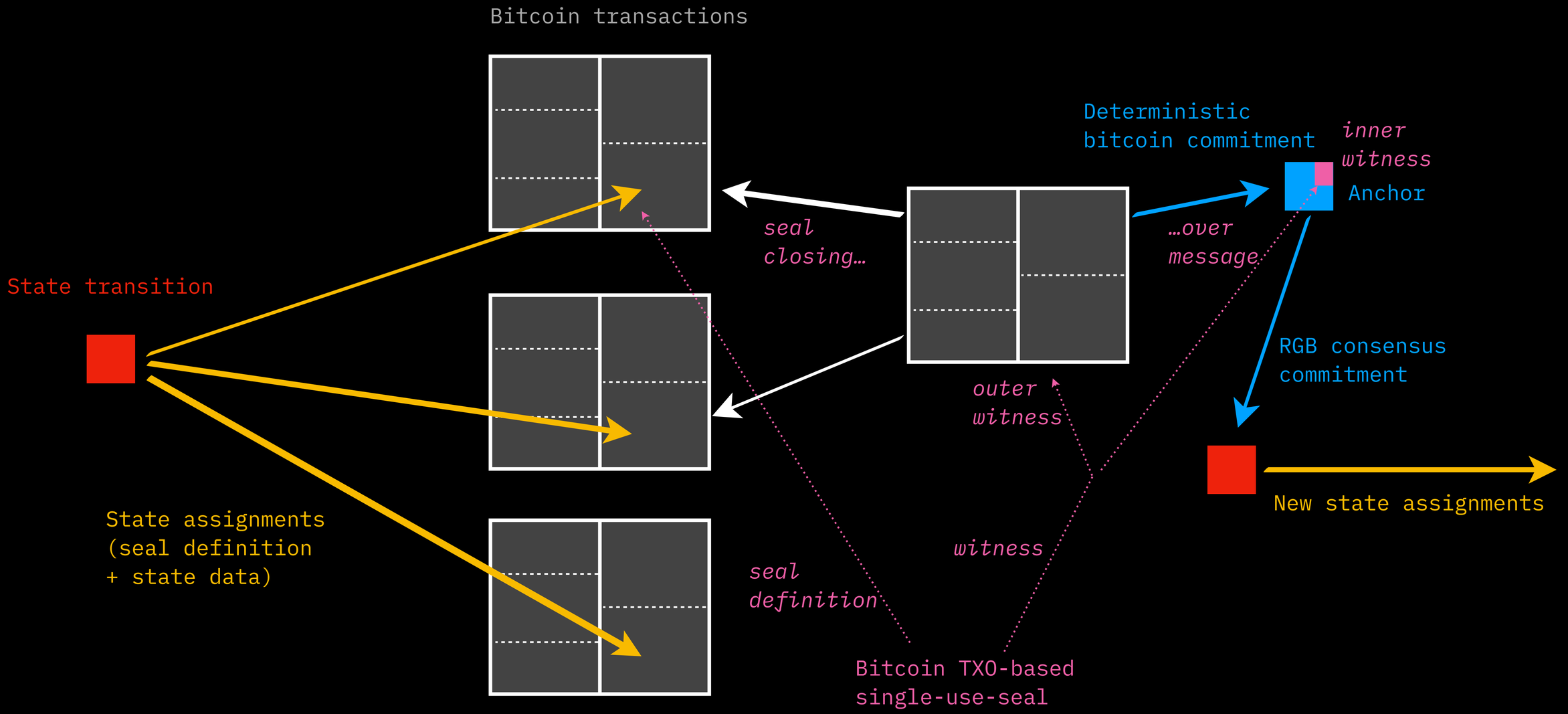
Contract nodes

- State types
  - State assignment
    - Seal definition
    - State data of specific type
- Metadata field
  - Metadata record
    - State data of specific type
- Simplicity scripts for additional validation of all future transitions starting from this one

Schema

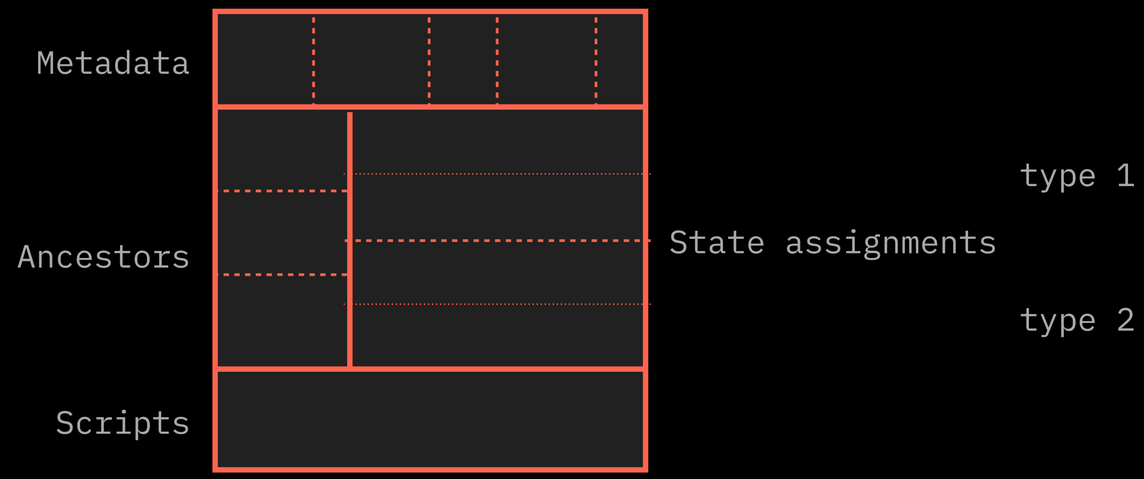
- Assignment type definitions
  - what data format may they have?
  - how the state should be validated? (isolation!)
- Metadata field type definitions
  - what data format may they have?
  - how the data should be validated?
- State transition type definitions
  - Which state type seals and how many of them the transition may close?
  - Which state it may define?
  - How many state assignments of each type are allowed?
  - What metadata can be present?
- What script extensions are allowed at each level

Spending TXOs with an assigned state



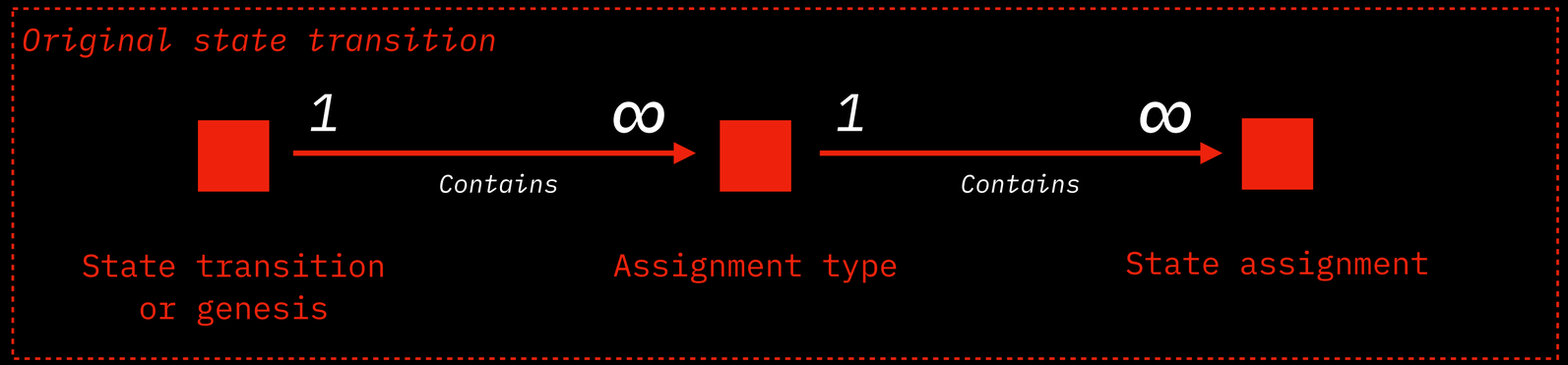
Smart contract node

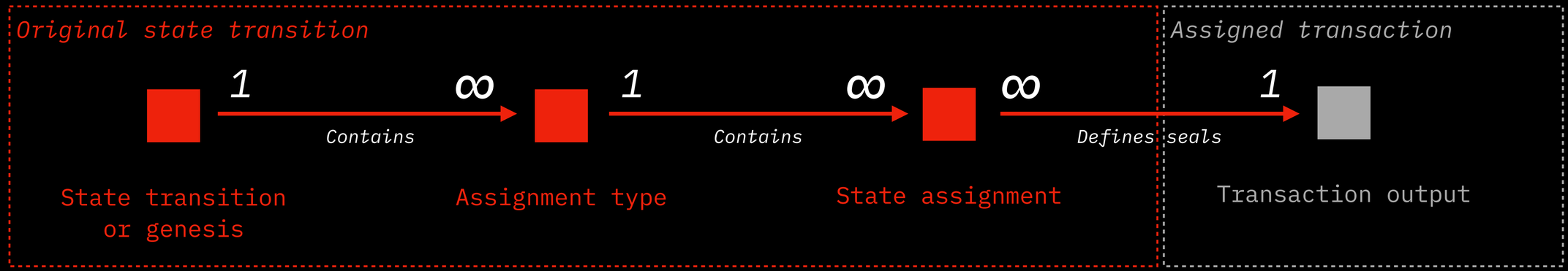
- Metadata
- State assignments
- Scripts
- Genesis:
  - Commitment to schema
  - Network
- State transitions:
  - Ancestors

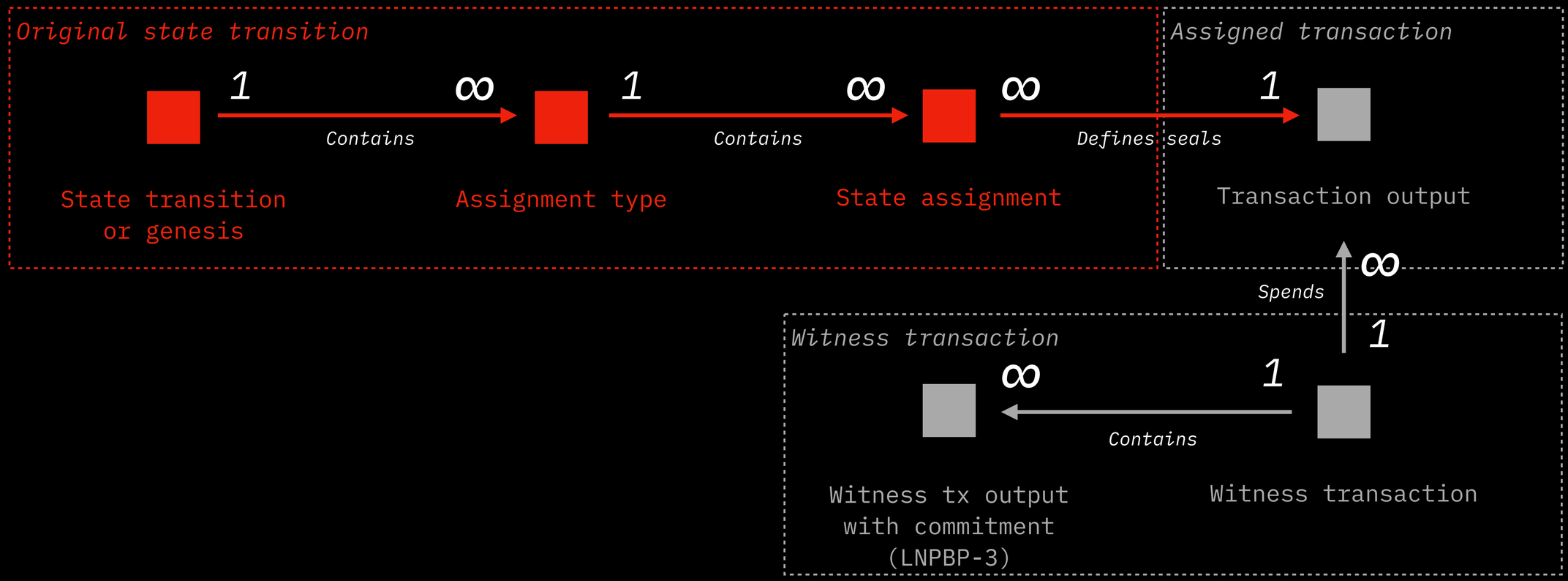


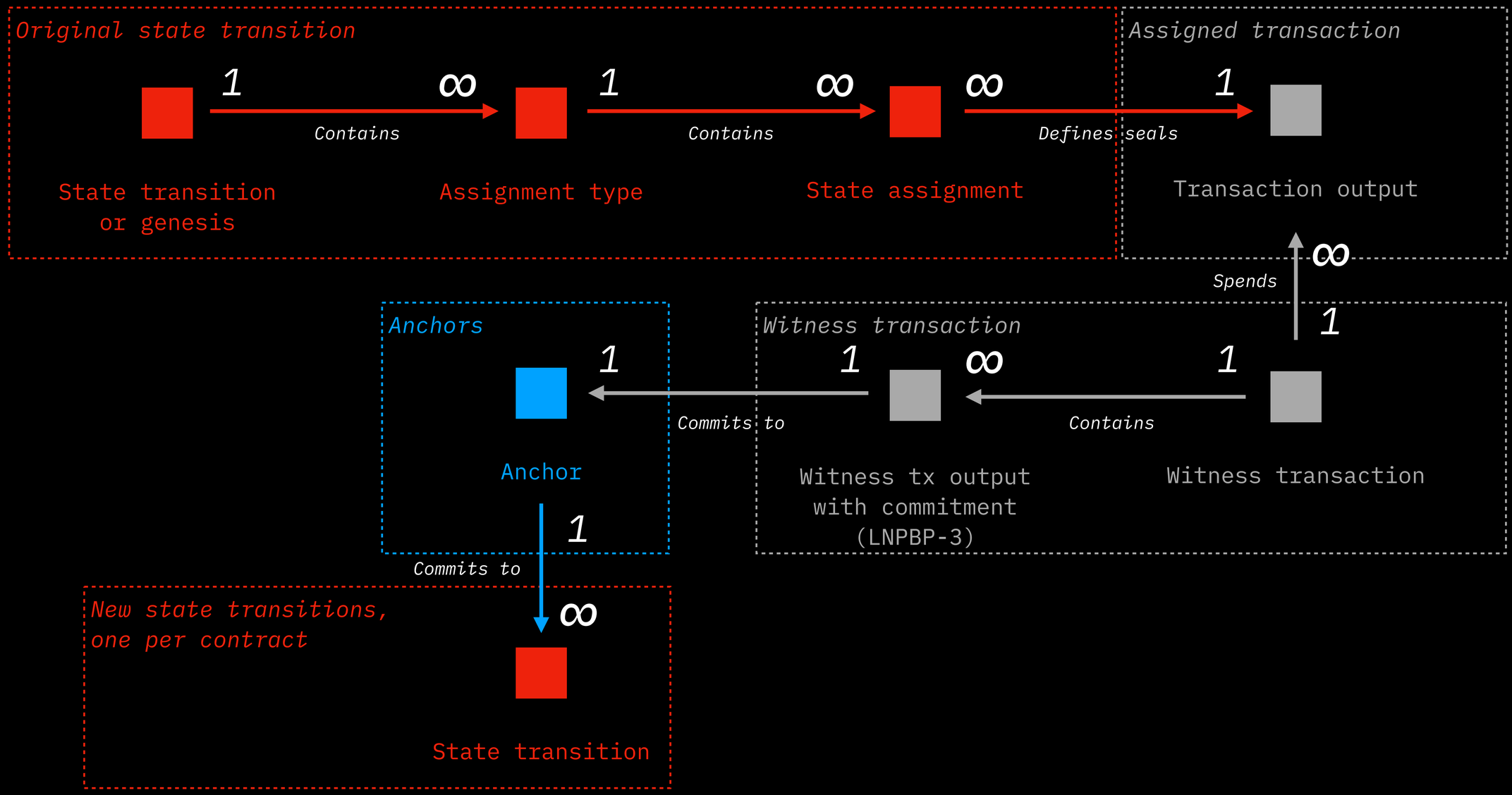
Why no version in RGB nodes?

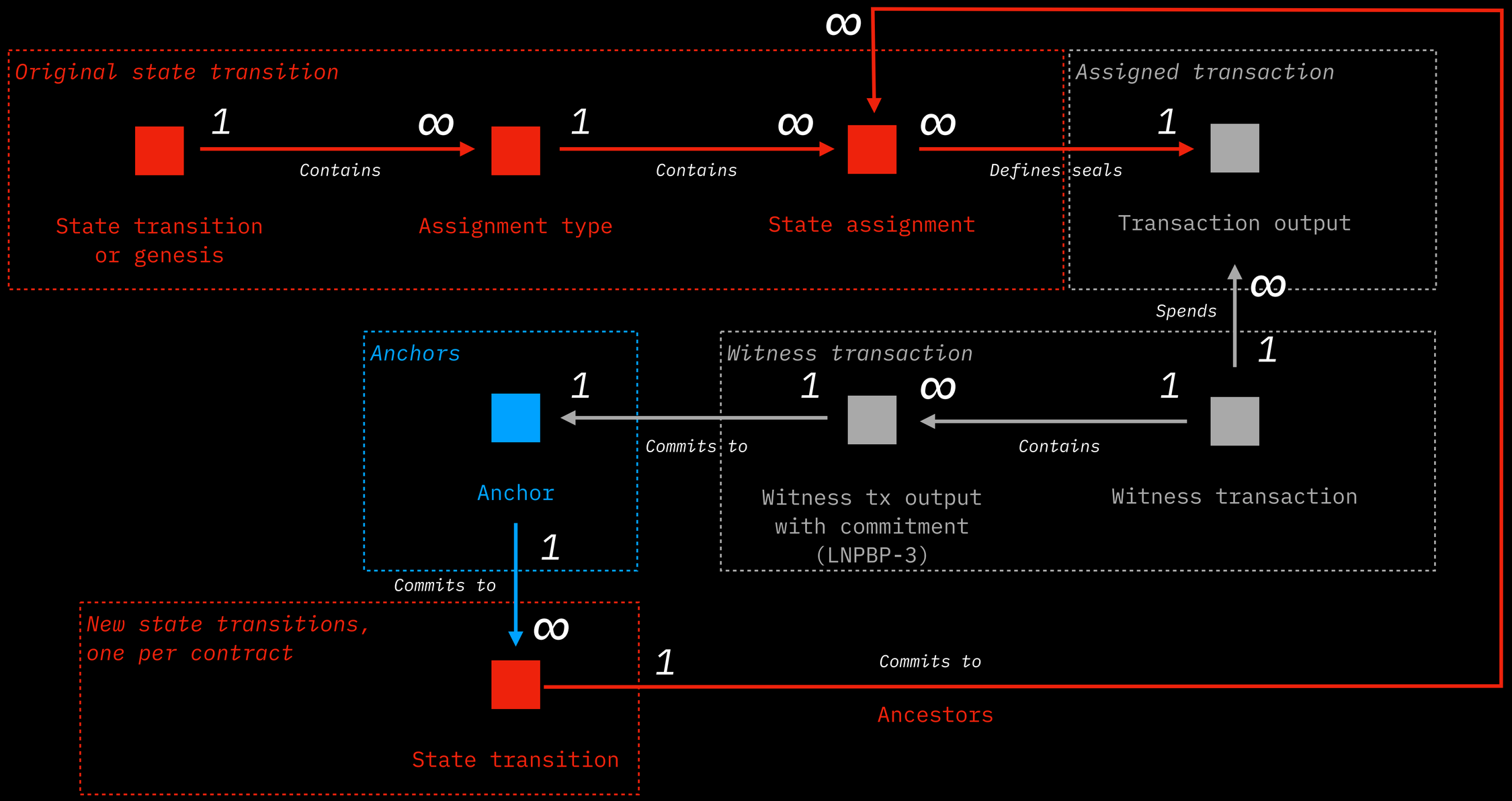
- The version can’t be changed within RGB Smart contract and is defined only by the issuer
- The version is defined by the Schema


Original state transition
1
∞
Contains
1
∞
Contains
State transition or genesis
Assignment type
State assignment

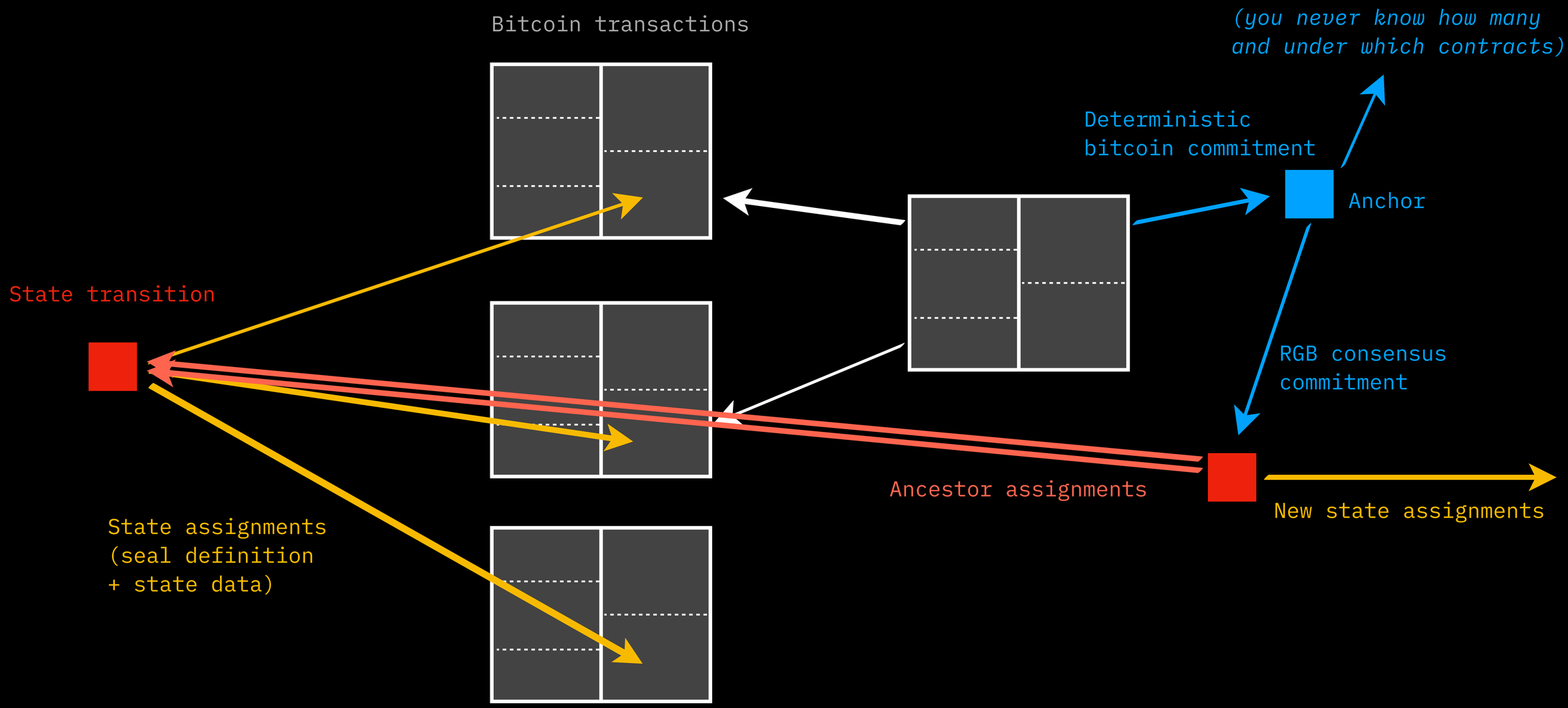

Original state transition
Assigned transaction
1
∞
Contains
1
∞
Contains
∞
1
Defines seals
State transition or genesis
Assignment type
State assignment
Transaction output

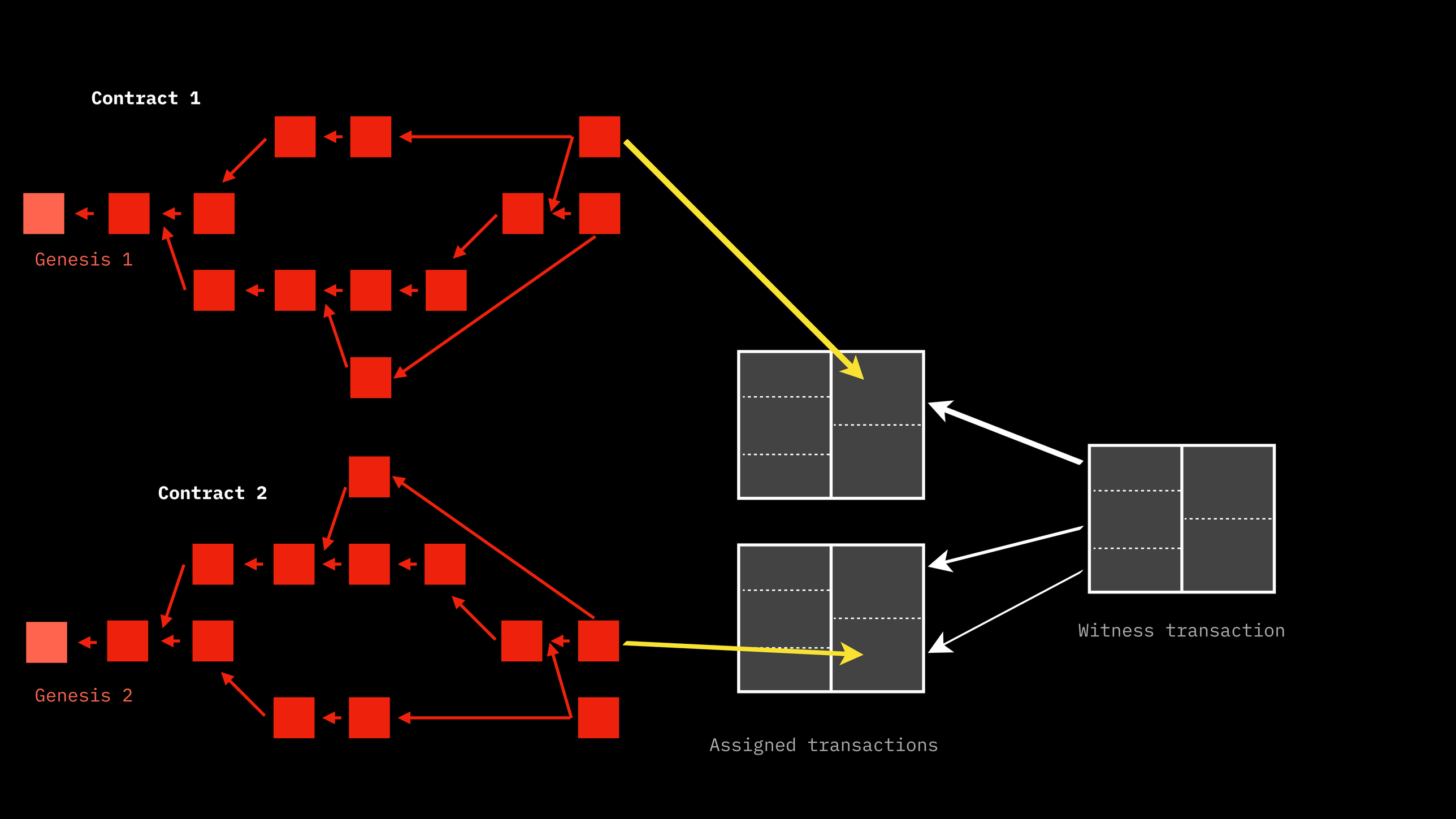

Original state transition
1
∞
Contains
1
∞
Contains
∞
1
Defines seals
Assigned transaction
State transition or genesis
Assignment type
State assignment
Transaction output
∞
Spends
1
Witness transaction
∞
1
Contains
Witness tx output with commitment (LNPBP-3)
Witness transaction

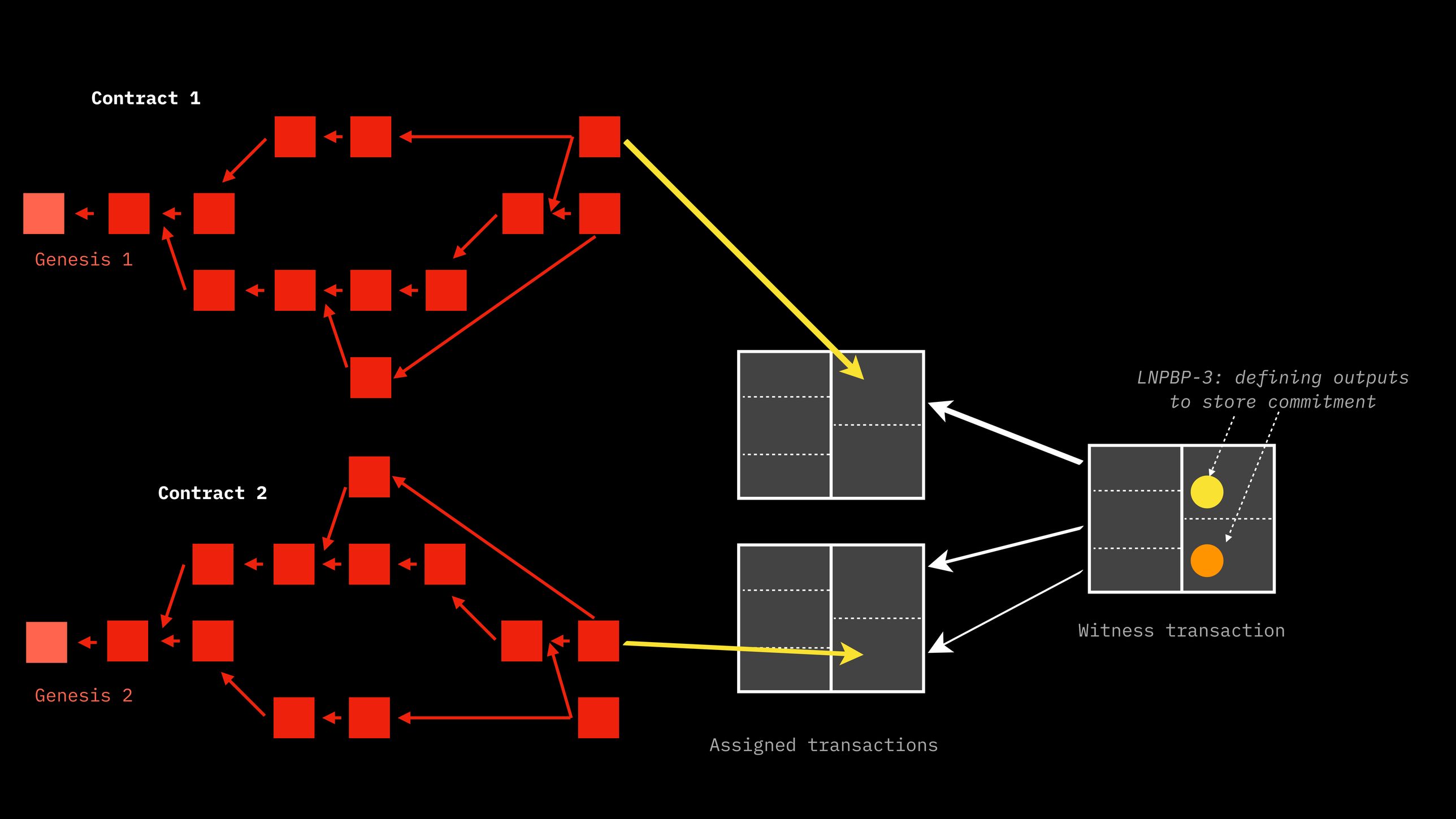

Original state transition
1
∞
Contains
1
∞
Contains
∞
Defines seals
1
Assigned transaction
State transition or genesis
Assignment type
State assignment
Transaction output
∞
Spends
1
Anchors
1
Commits to
Witness transaction
1
∞
1
Contains
Anchor
Witness tx output with commitment (LNPBP-3)
Witness transaction
1
Commits to
∞
New state transitions, one per contract
State transition

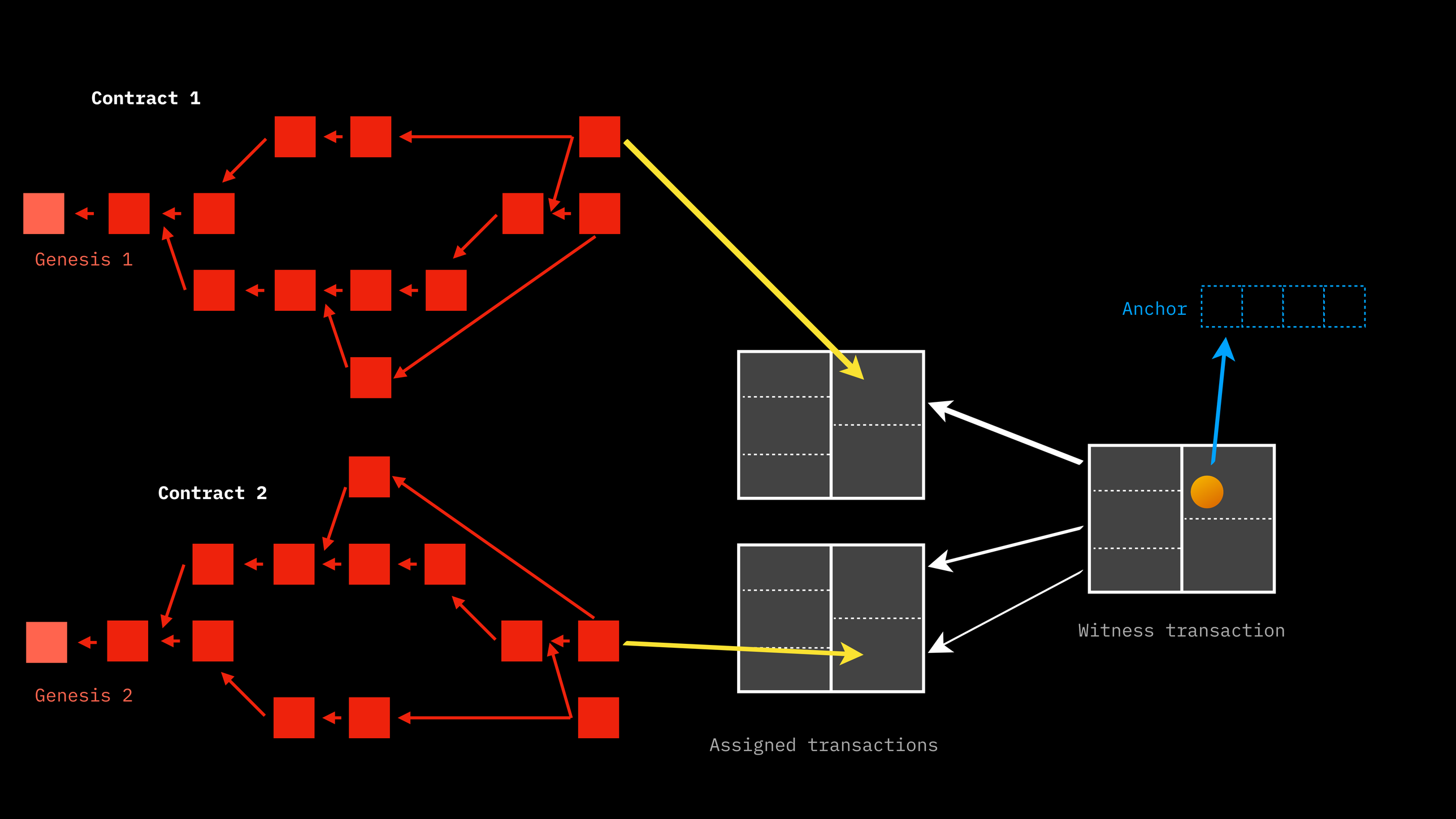

Original state transition
1
∞
Contains
1
∞
Contains
∞
Defines seals
1
Assigned transaction
State transition or genesis
Assignment type
State assignment
Transaction output
∞
Spends
1
Anchors
1
Commits to
Witness transaction
1
∞
1
Contains
Anchor
Witness tx output with commitment (LNPBP-3)
Witness transaction
1
Commits to
New state transitions, one per contract
∞
1
Commits to
Ancestors
State transition

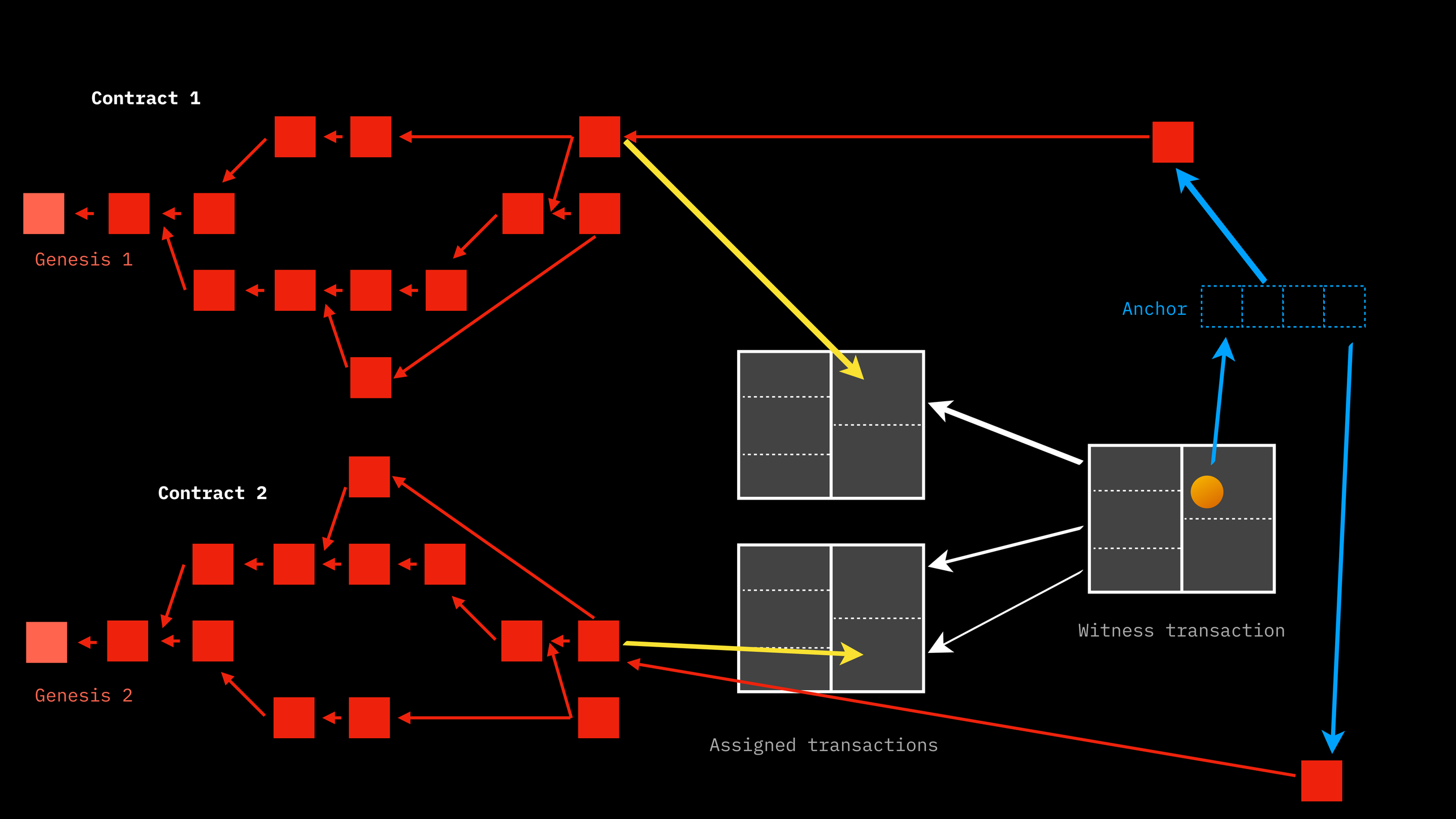
Spending TXOs with an assigned state




Contract 1
Genesis 1
Contract 2
Genesis 2
Assigned transactions
Witness transaction


Contract 1
Genesis 1
Contract 2
Genesis 2
LNPBP-3: defining outputs
to store commitment
Witness transaction
Assigned transactions


Contract 1
Genesis 1
Contract 2
Genesis 2
Anchor
Witness transaction
Assigned transactions


Contract 1
Genesis 1
Anchor
Contract 2
Witness transaction
Genesis 2
Assigned transactions

Transitions, Transactions & Anchors

- Bitcoin transactions:
  - Assignment (contains one or many outputs with assigned state)
  - Witness (spends one or many output with assigned state)
- State transition -> assignment type -> state assignment
  - Points to many ancestors per each type
  - Assigns multiple states of multiple types to multiple tx outputs (seal definitions)
- Anchor
  - Links transactions with transitions
  - Many per witness transaction, one per witness transaction output
  - May link one output to many state transitions
- Still, each witness transaction can be linked with a single state transition per RGB smart contract