RGB Tech Internals, part II

Data Structures Hierarchy

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

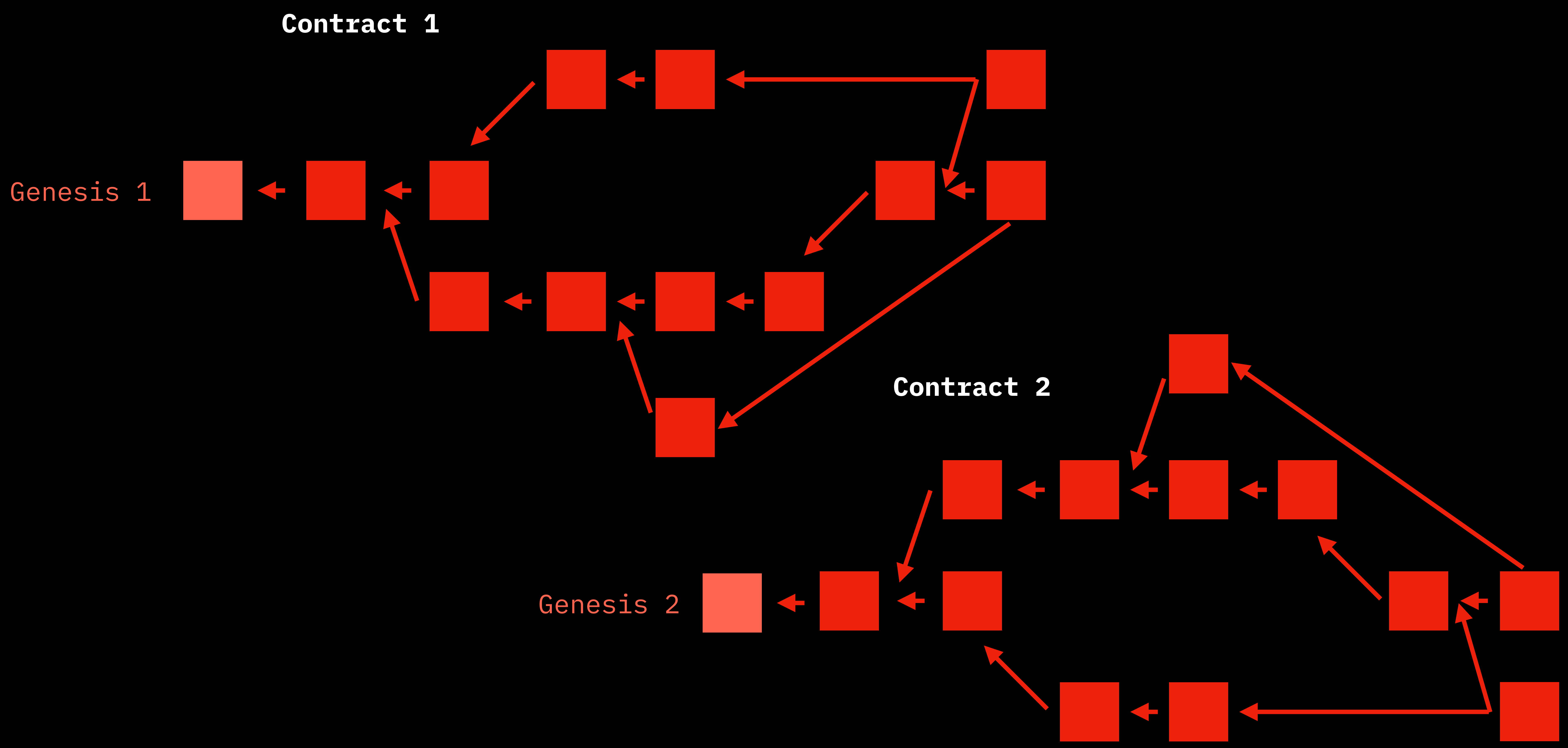
RGB Tech Internals Parts

Our RGB definition

Smart contracts layer on top of Bitcoin & Lightning Network, with emphasise on privacy & scalability

“Mass market” definition

RGB client-side view of multiple contract DAGs



🟥 State transition

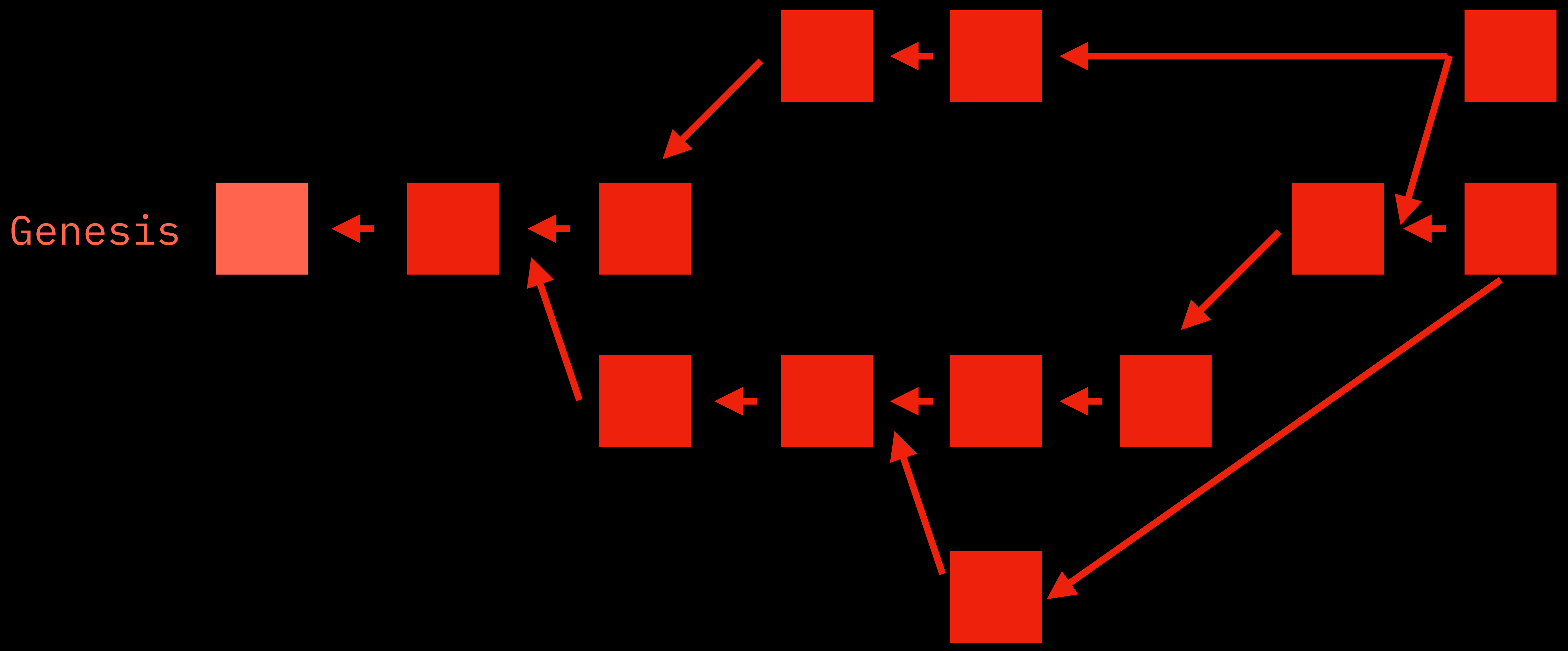
RGB contracts are a sort of data “shards”

However, different RGB contracts (i.e. state histories with different genesis) never interact

There is special way of preventing this interaction at the level of single-use-seals and bitcoin commitments called **anchors**

We will go into how they work later; for now let’s dig into a single RGB contract details

Single RGB contract history



🟥 State transition

Contract data structures in LNP/BP Core Library

- Contract Node (trait **Node**)
  - Genesis (concrete type)
  - State transition (concrete type)

- **rgb::contract** mod constrains this data structure definitions + all nested data types
  - seal definitions
  - state data
  - state assignments to seals
  - metadata fields

Contract data identifiers

Tag-hashed (taproot-like) consensus-serialized data of the node:

- **ContractId**: only for Genesis, represents unique id for the whole RGB contract history
- **NodeId**: can be generated for both Genesis and State Transition

While these ids are equivalent at the byte level, they are different data types in order to separate semantics of their use

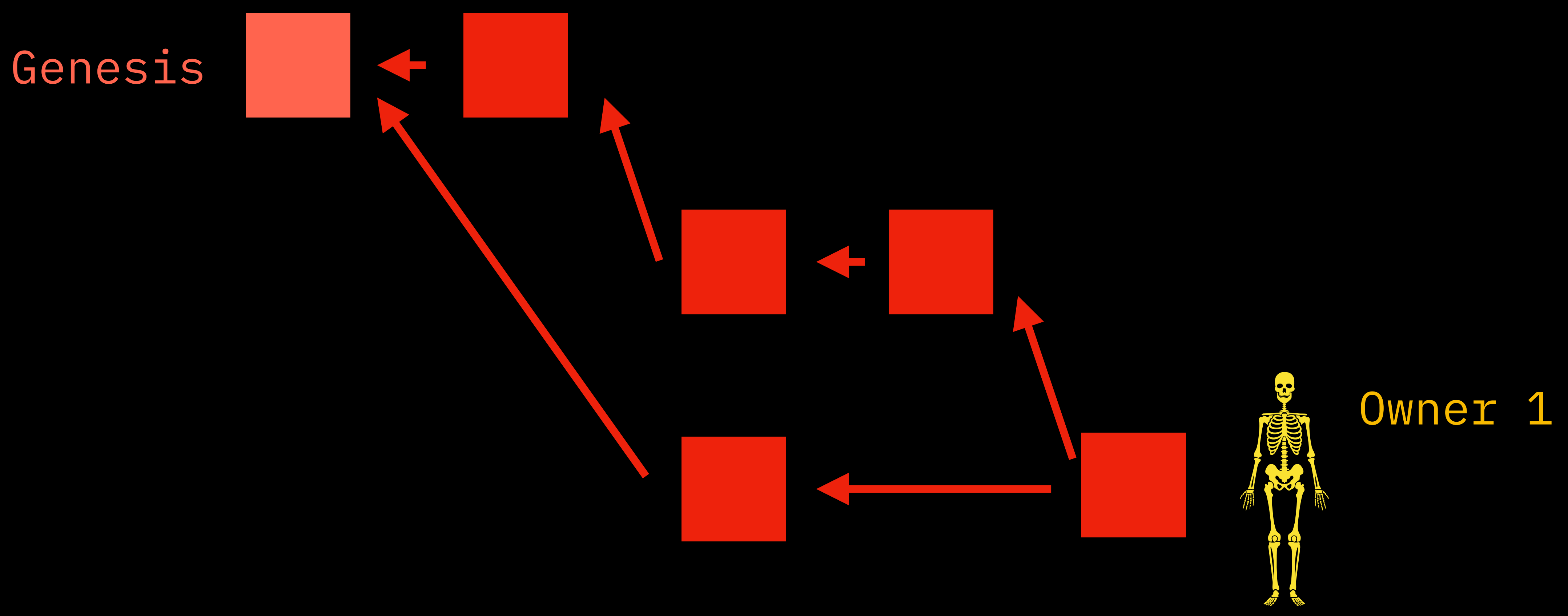
What makes RGB special?

Privacy

In most of the cases nobody will have a complete history of some RGB contract

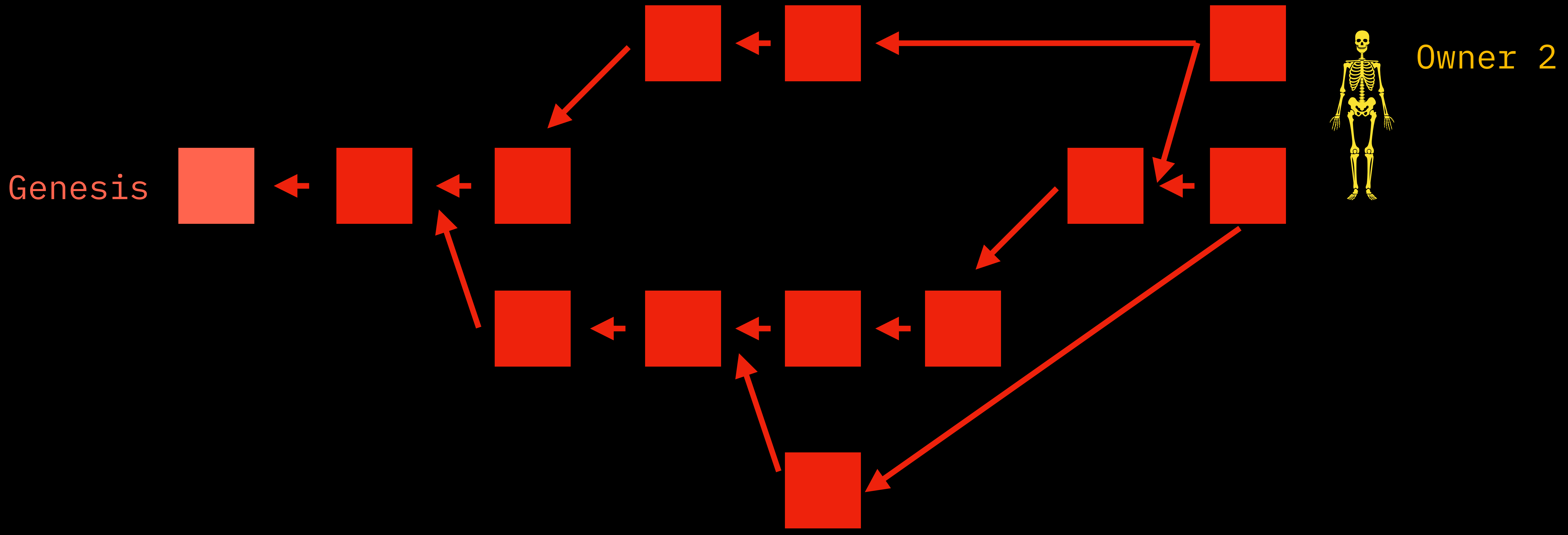
While RGB contracts are isolated “shards”, within each contract the history is also split into shards, this time owner-specific

Owner 1 perspective



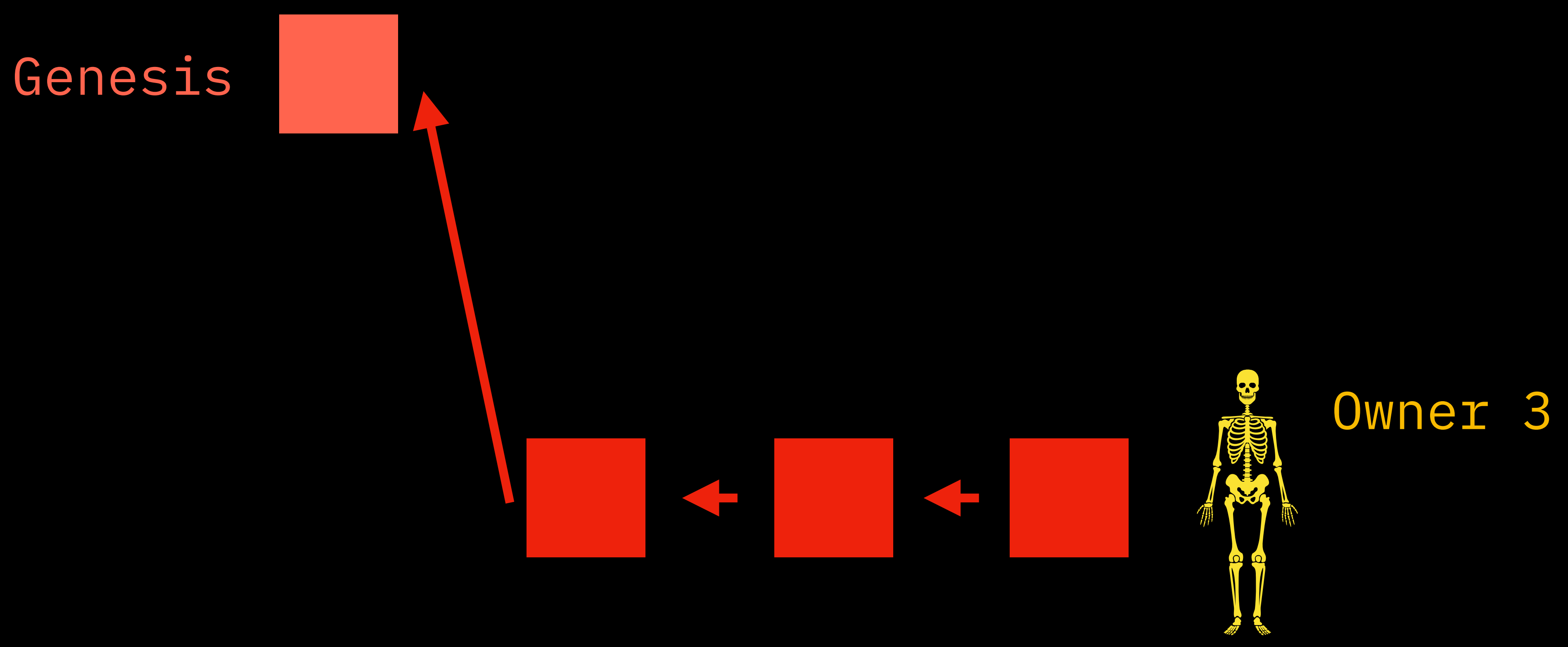
🟥 State transition

Owner 2 perspective



🟥 State transition

Owner 2 perspective without shared part



🟥 State transition

This is a feature, not a bug

- Increases privacy
- Allows scalability

State transitions may “forget” their data

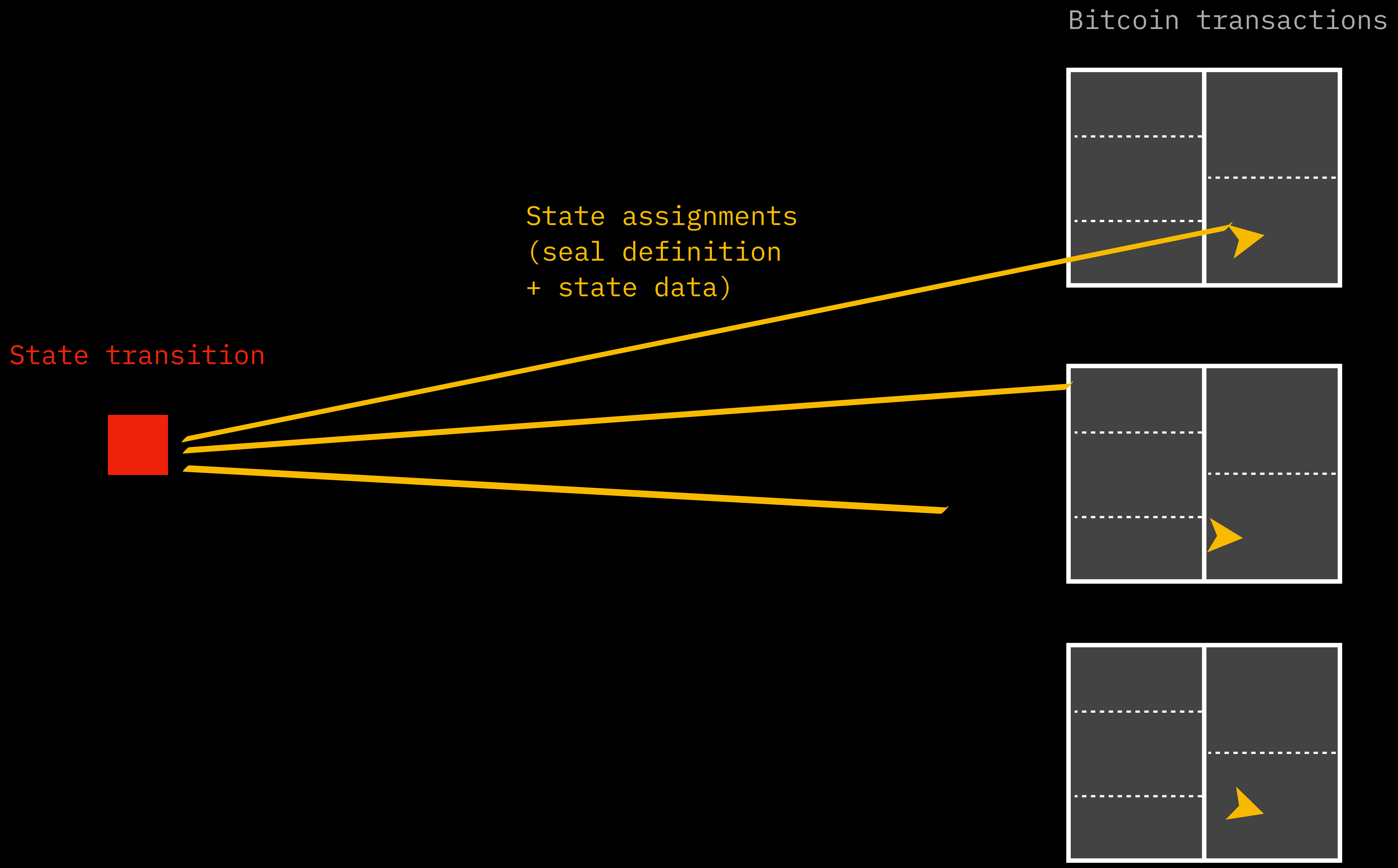
- If some part of transition data are not required for the future validation, it may be hidden beyond commitment, with the actual data being deleted forever and not accessible by the future owners
- The future owners still will be able to validate the history
- This is called a **conceal procedure** automatically performed by the core library during consignment preparation
- So even shared state transition may differ in their revealed/known information from different owners perspective

What RGB is?

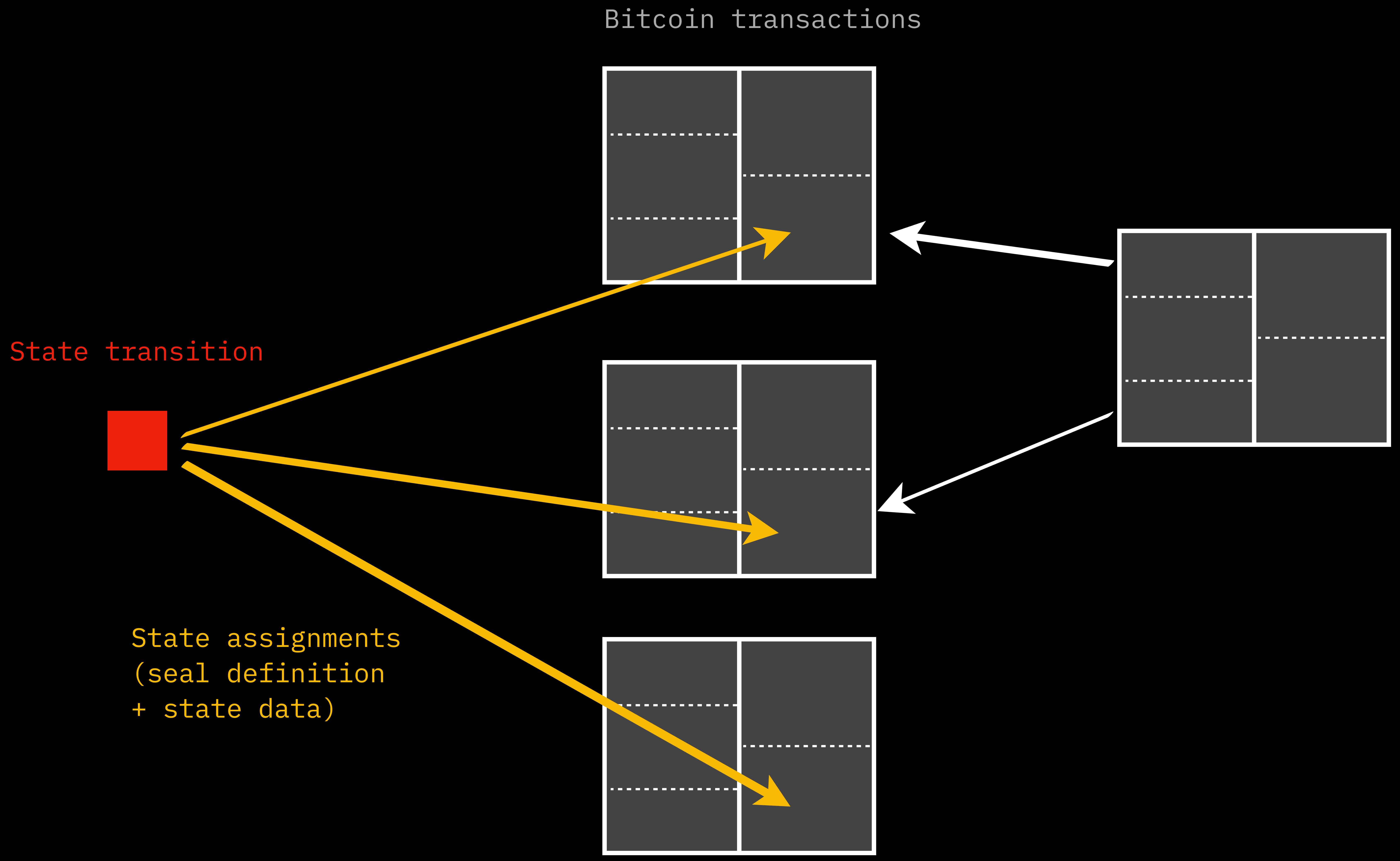
Distributed system of partially-replicated state machines without globally-known state, having nearly-synchronous state consistency property enforced by consensus protocol of underlying layer (single-use-seal medium).

Computer Science definition

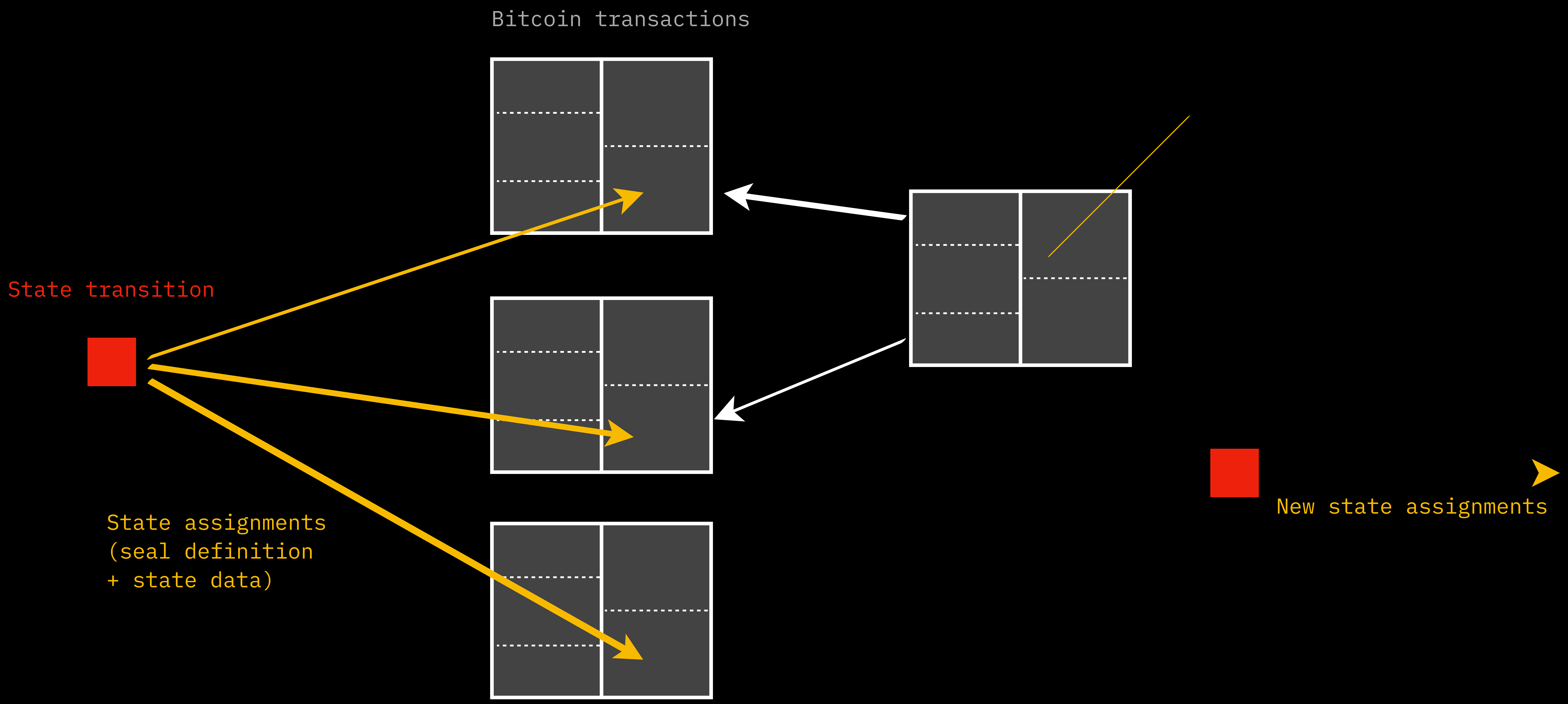
From transition to transactions: one-to-many



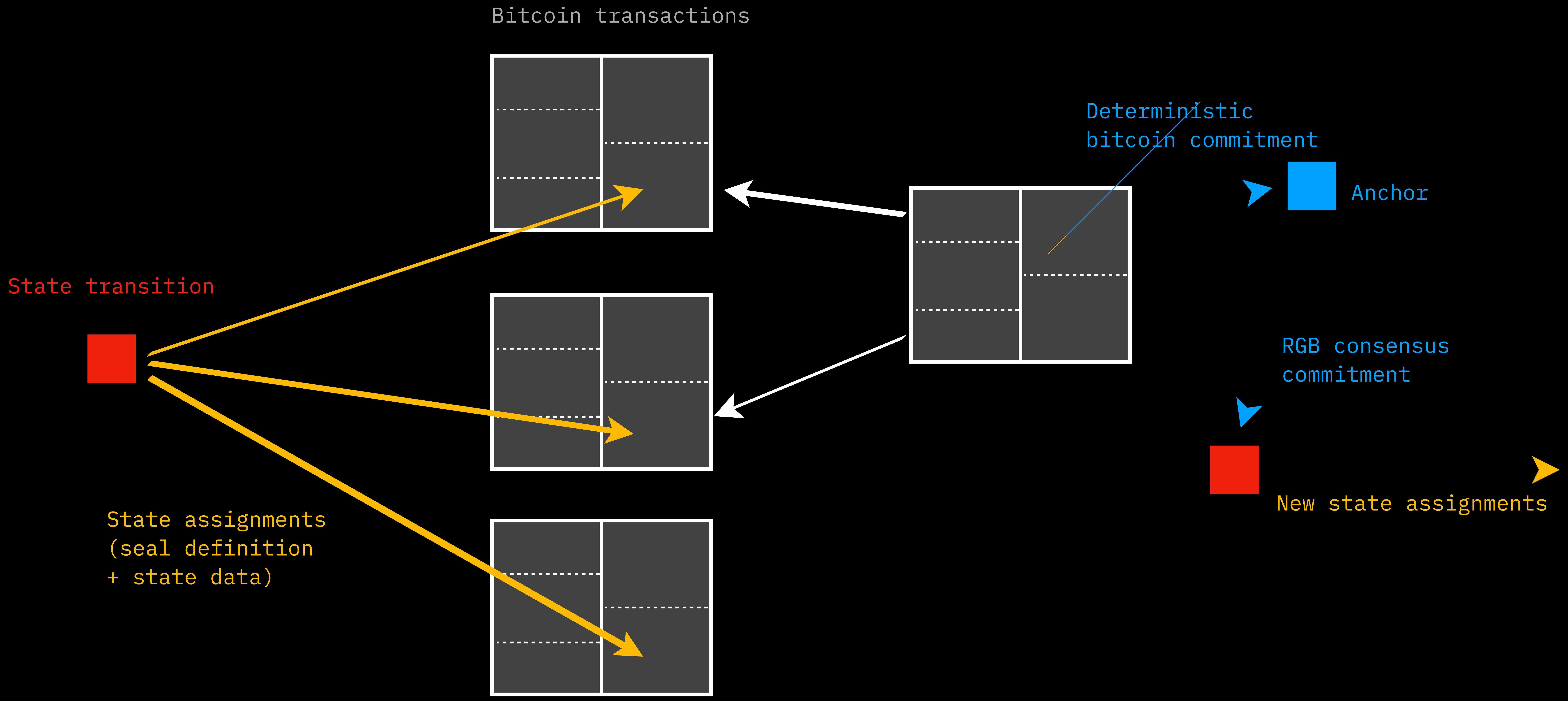
Spending TXOs with an assigned state



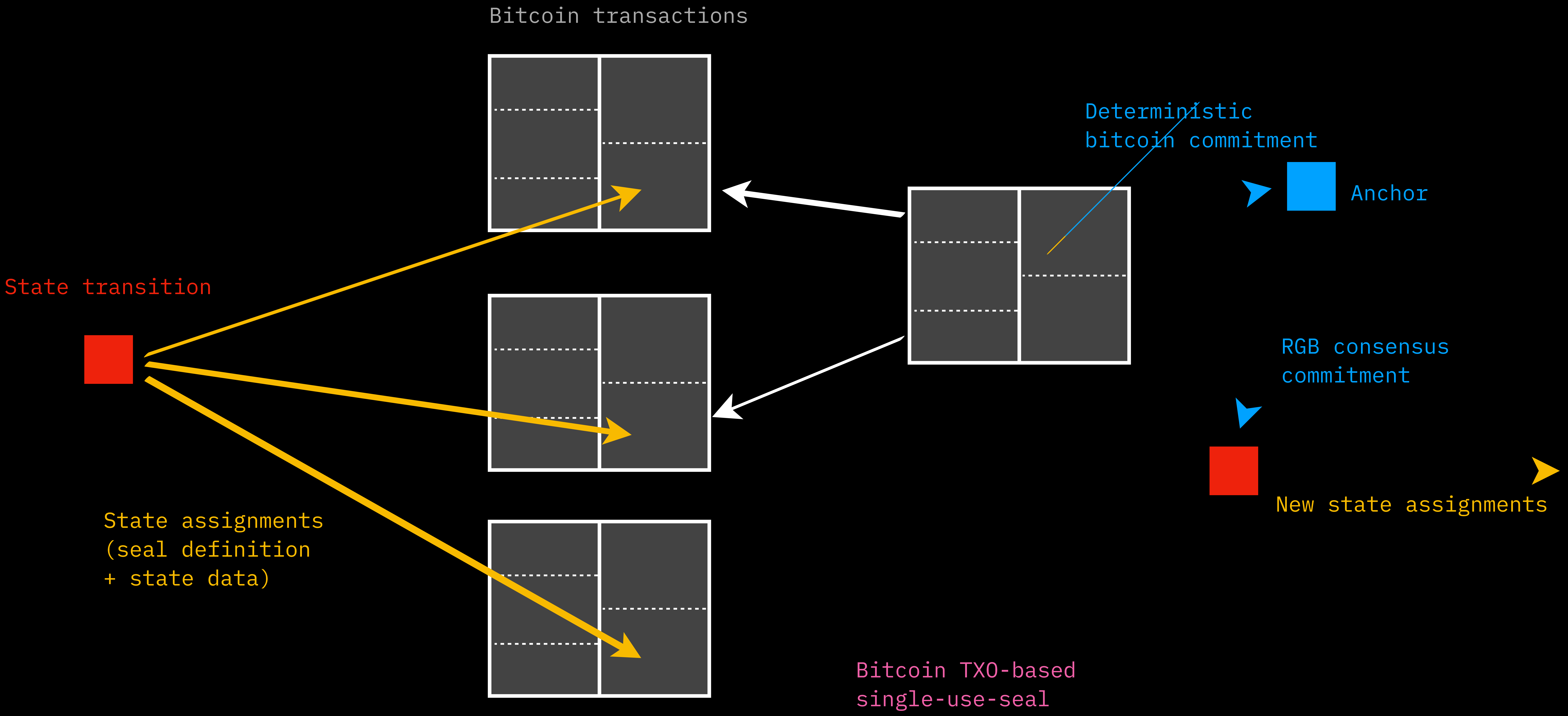
Spending TXOs with an assigned state



Spending TXOs with an assigned state



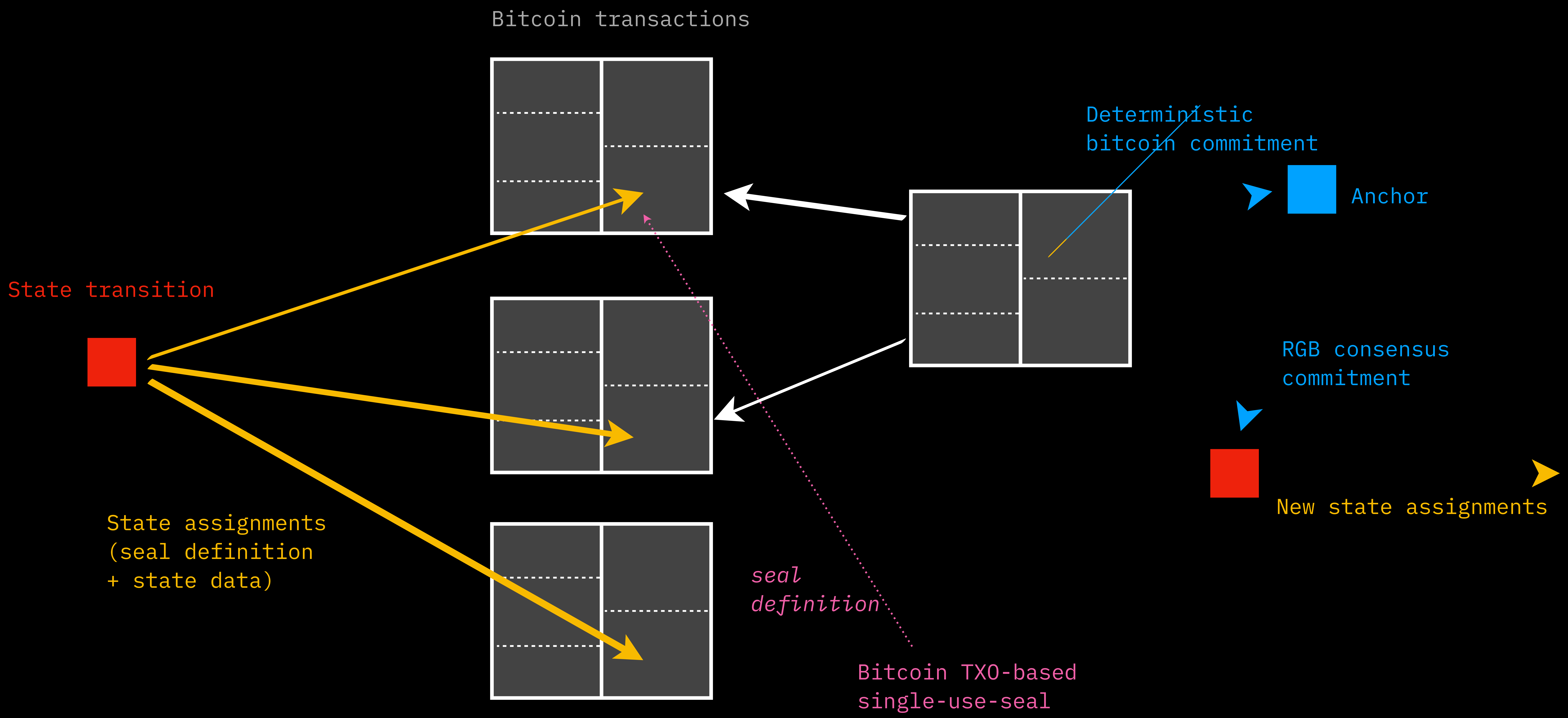
Spending TXOs with an assigned state



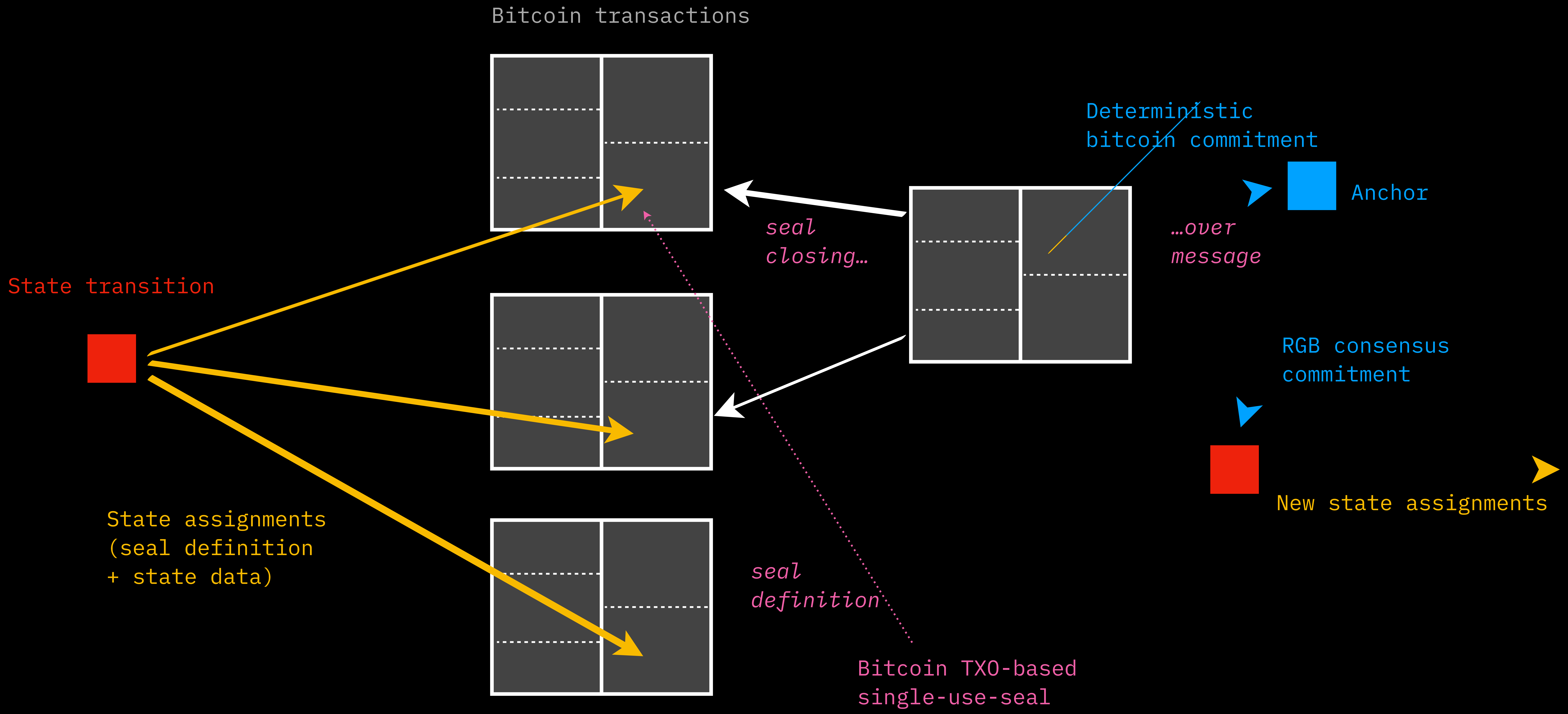
Single-use seal

- Two parties (Alice, Bob), three methods:
  - seal <- Define()
    done by Alice, accepted by Bob
  - witness <- Close(seal, message)
    close a seal over a message, done by Alice
  - true | false <- Verify(seal, witness, message)
    verify that the seal was closed, done by Bob

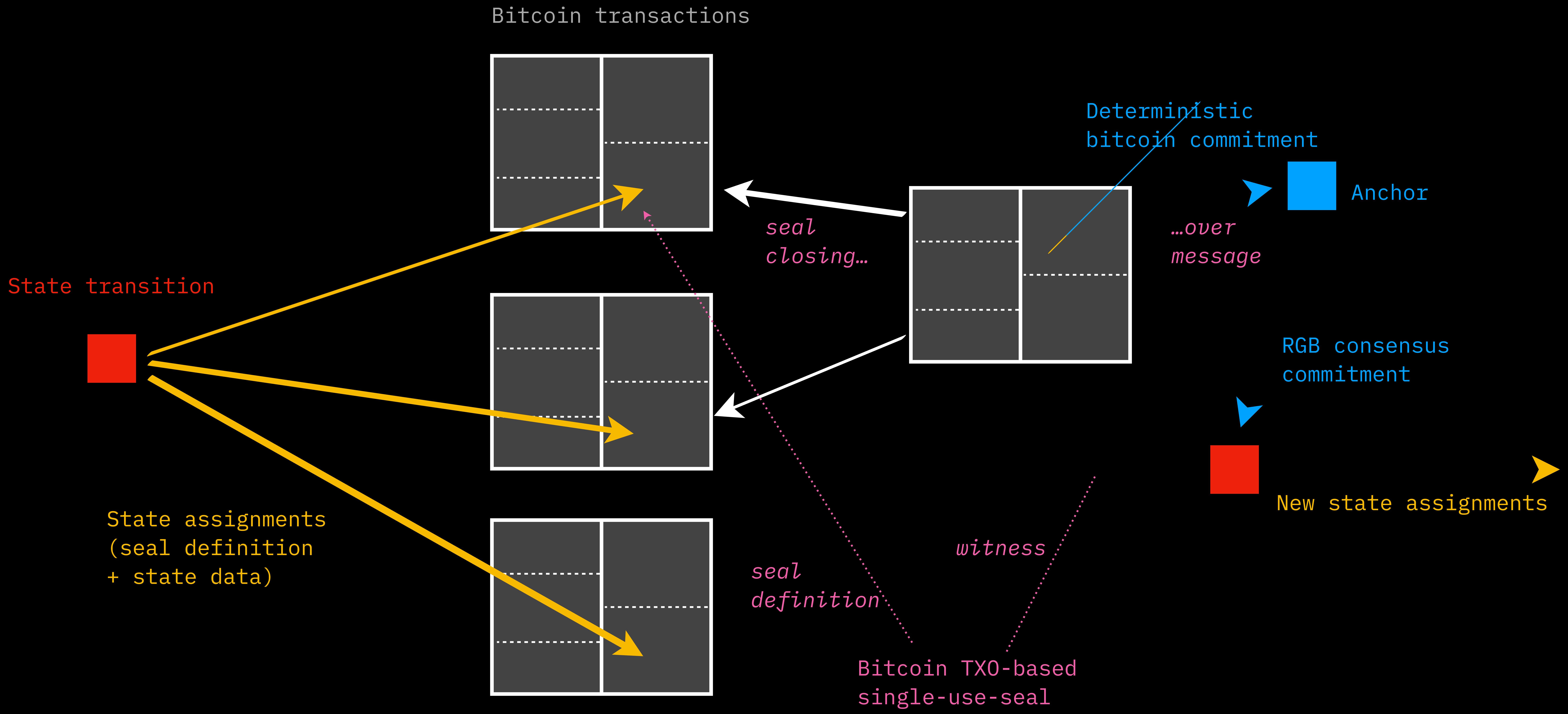
Spending TXOs with an assigned state


Bitcoin transactions
Deterministic bitcoin commitment
Anchor
State transition
RGB consensus commitment
New state assignments
State assignments (seal definition + state data)
seal definition
Bitcoin TXO-based single-use-seal

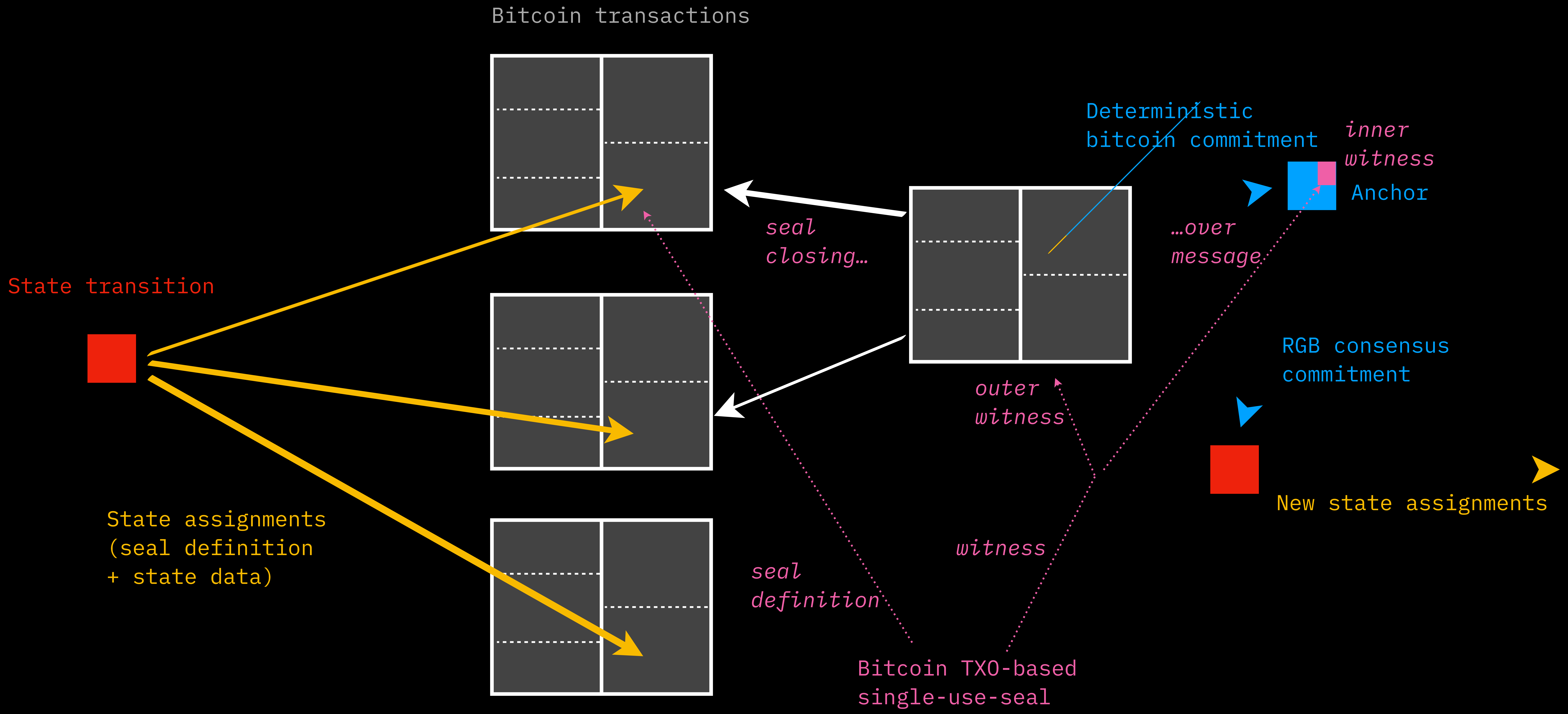
Spending TXOs with an assigned state



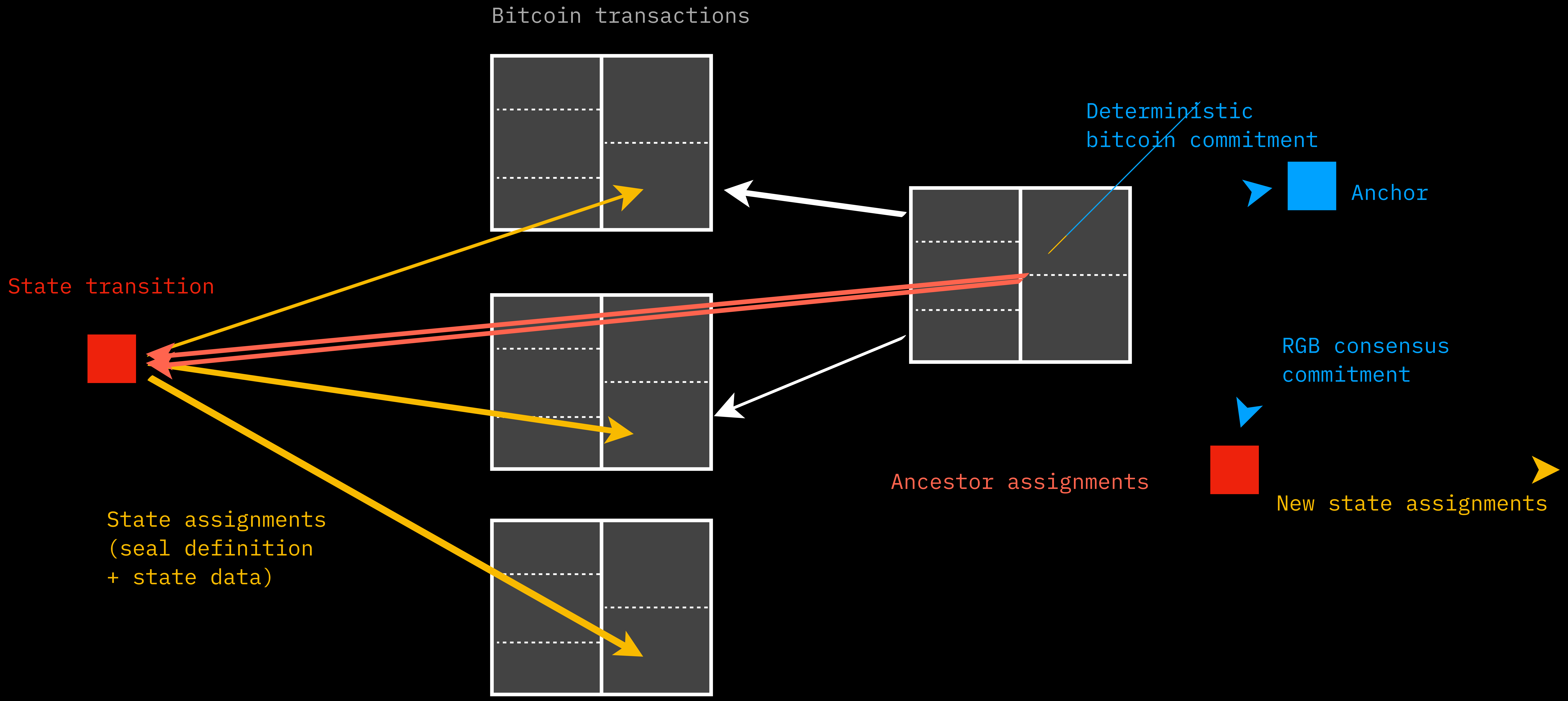
Spending TXOs with an assigned state



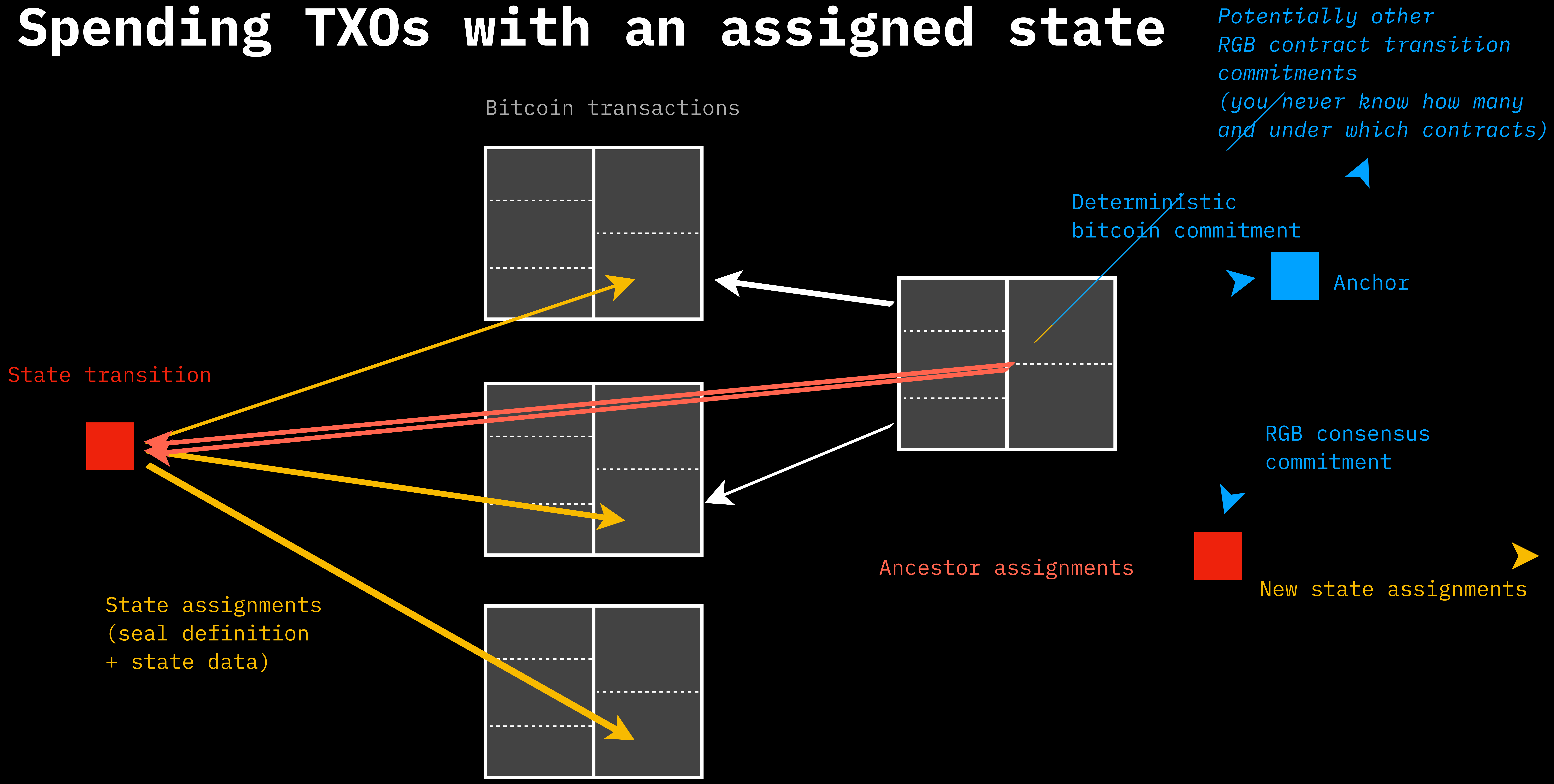
Spending TXOs with an assigned state



Spending TXOs with an assigned state



Spending TXOs with an assigned state


Potentially other RGB contract transition commitments (you never know how many and under which contracts)
Bitcoin transactions
Deterministic bitcoin commitment
Anchor
State transition
RGB consensus commitment
Ancestor assignments
New state assignments
State assignments (seal definition + state data)

High-level data structures

- As an owner, you can own state from multiple RGB contracts with their arbitrary-complex DAGs of past history
- They may cross-interact via anchors in a very complex manner at the level of bitcoin transactions
- Altogether, these data is kept in a single binary storage called **stash**
- Program never interacts with the stash data directly
- And even inside the stash different contracts' data is very well separated from each other

High-level data structures

- When you need to transfer an ownership over some new state you create partial binary extract from the stash, named **consignment**
- Consignment always “hides” all confidential data that the new owner does not need to know (with the conceal procedure)
- New owner validates consignment
- When the transaction committing to the last anchor in consignment is mined
  - New owner *merges* consignment it into its own stash
  - Old owner removes unnecessary data from its own stash with the *forget* procedure

Stash data structures in LNP/BP Core Library

- **rgb::stash** mod constrains this data structure definitions + all nested data types
  - **Stash**: a trait used by specific RGB node implementation
  - **Consignment**: concrete data structure
  - **Anchor**: concrete structure orchestrating all commitment and single-use-seals procedures
  - **Disclosure**: a way to disclose some private information about the owned state from stash, if needed (like the issuer disclosing additional data on some of his operations)

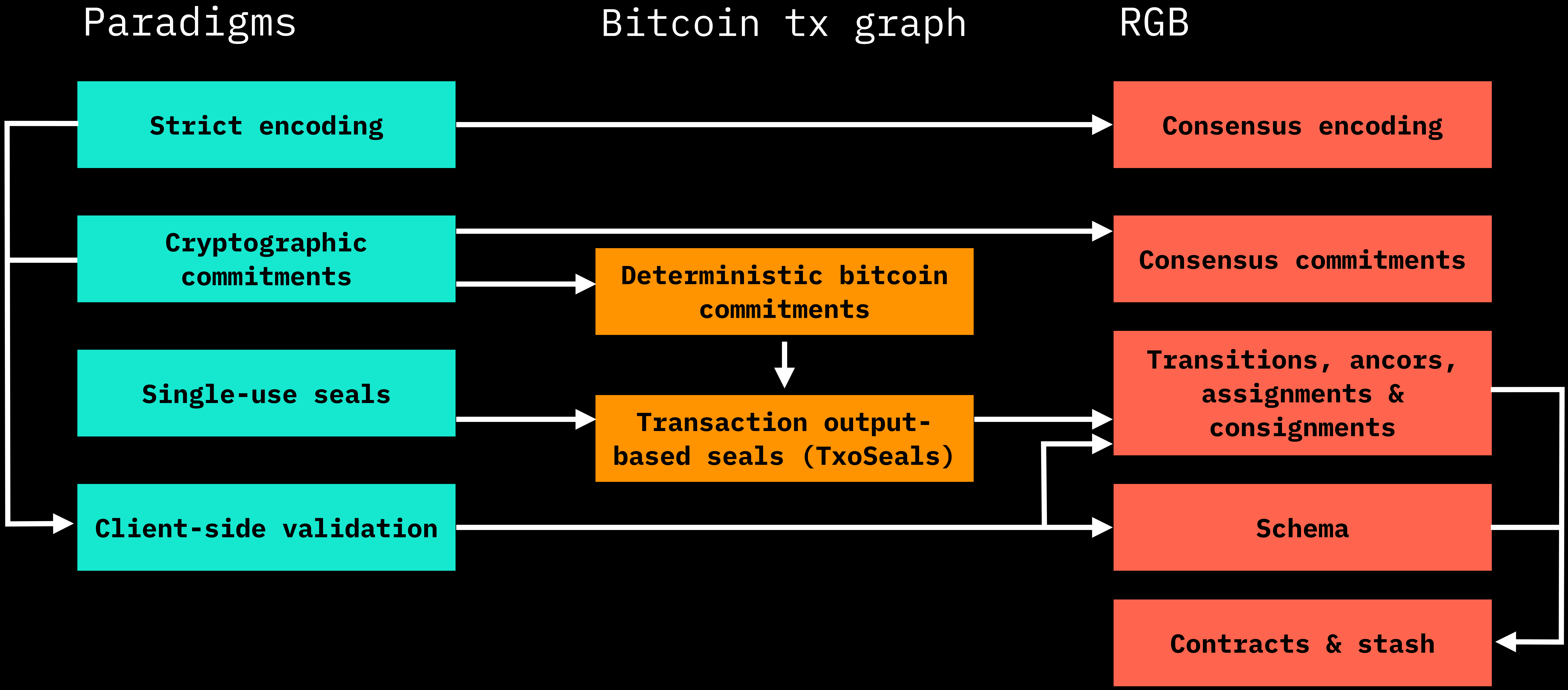
Overview of RGB mod in LNP/BP Core Library

- **rgb::contract**: everything related to a single RGB contract and its DAG outside of bitcoin transaction scope
- **rgb::stash**: multi-contract RGB data and linking to bitcoin transactions
- **rgb::schema**: RGB schema that can be used by multiple contracts (will be presented in the next presentation)
- **rgb::vm**: scripting extensions (will be presented in the next presentation)
- **rgb::validation**: top-level logic orchestrating validation process involving all above mentioned structures

Underlying layers of LNP/BP Core Library

- **bp::dbc**: deterministic bitcoin commitments
  - based on **paradigms::commit_verify** - streamlined workflow for robust cryptographic commitments of any sort
- **bp::seals**: bitcoin-specific TXO-based implementation of single-use-seals
  - based on **paradigms::single_use_seals** - generic single-use-seals APIs
  - uses deterministic bitcoin commitments
- **lnpbps::lnpbp4**: Multi-message commitments used by anchors to hide & mix RGB contract information
- **paradigms::strict_encoding** and **bp::strict_encoding**: specially-designed encoding standards used for consensus-critical serialization and commitments (including single-use-seals and deterministic bitcoin commitments)

From paradigms through bitcoin to RGB


Paradigms
Bitcoin tx graph
RGB
Strict encoding
Cryptographic commitments
Single-use seals
Client-side validation
Deterministic bitcoin commitments
Transaction output-based seals (TxoSeals)
Consensus encoding
Consensus commitments
Transitions, ancors, assignments & consignments
Schema
Contracts & stash

Hell of a lot of terminology: RGB

Contracts

- Node & NodeId
- Genesis & ContractId
- State transition
- State assignment
- Assignment variant
- State type
- State data
- Metadata & metadata fields
- Data type

Schema & Scripts

- Schema
- Transition type
- Field type
- Assignment type
- Occurrence boundaries
- Bit dimensions
- Script library
- ABI
- Script extensibility

Stash

- Anchor
- Stash
- Consignment
- Disclosure
- Consign procedure
- Conceal procedure
- Validate procedure
- Merge procedure
- Forget procedure

Hell of a lot of terminology: underlying layers

Deterministic bitcoin commitments

- Container
- Commitment
- Commitment embedding
- Proof
- Supplement
- Protocol-specific entropy
- Container construction/ deconstruction
- LockScript

Single-use-seals

- Seal definition
- Seal closing over message
- Witness
  - inner witness
  - outer witness
- Seal blinding
- Multimessage commitments

Encodings

- Strict encoding
- Commitment encoding
- Merklization commitment encoding
- Storage encoding
- Pedersen commitments
- Blinding factors
- Bulletproofs
- Conceal