Conference follow-up

What has changed during the conference

- Reworked public key tweaking (LNPBP-1)
- Rejected current confidential multi-message commitments (LNPBP-4) in its Pedersen-commitment part
- Proposal to remove range proofs for confidential amounts and make them u256 < p (or <n?)
- Single-asset LN channels instead of multi-asset channels
- Simplicity script field with zero-length by default
- LN channel RGB funding with external transaction

Other LN-related updates

- Fees for onion routing must be valued in channel asset
- Do not change update_add_htlc, pass the required information inside the onion package, which is used even for direct transfers
- Decide on base_fee_rate
  - Make it always 0
  - Include into the channel_update message
- Consider Perun channels

Proposals emerged during the conference

- Taproot: possible use of length-extension attack mitigation of it with HMAC
- Confidential Assets: consider using u256 amounts and removing range proofs

New developments after the conference

- Bloom filters for LNPBP-4 instead of Pedersen commitments
- Different commitment serialisation algorithm for state transitions
- Kaleidoscope as a command-line RGB wallet
- Genesis state commitment is not published onto blockchain

Still opened questions

- Can we get rid of range proofs for confidential amounts?
  (need review by Andrew Poelstra)

- What is the best deterministic algorithm for script commitments?
  (miniscript footprints; stack emulation; homomorphic commitments)

- Is the new LNPBP-4 safe?
  (needs review by the community)

- When Simplicity? The roadmap.
  (question to Adam Back)

TODOs (hopefully before the New Year)

- On-chain RGB asset issuance and transfer with command-line wallet (kaleidoscope), which requires...
  - ...finalisation of rust-lnpbp libraries
  - ...work on rust-bitcoin: *nearly complete*
  - ...work on rust-wallet
- CI & DevOps: *done*
- Updating LNPBP standards with all discussed improvements and preparing them for community review round
- Approaching Andrew Poelstra to discuss CAv2 and RGB interoperability & removing range proofs for u256-amounts

New developments for software

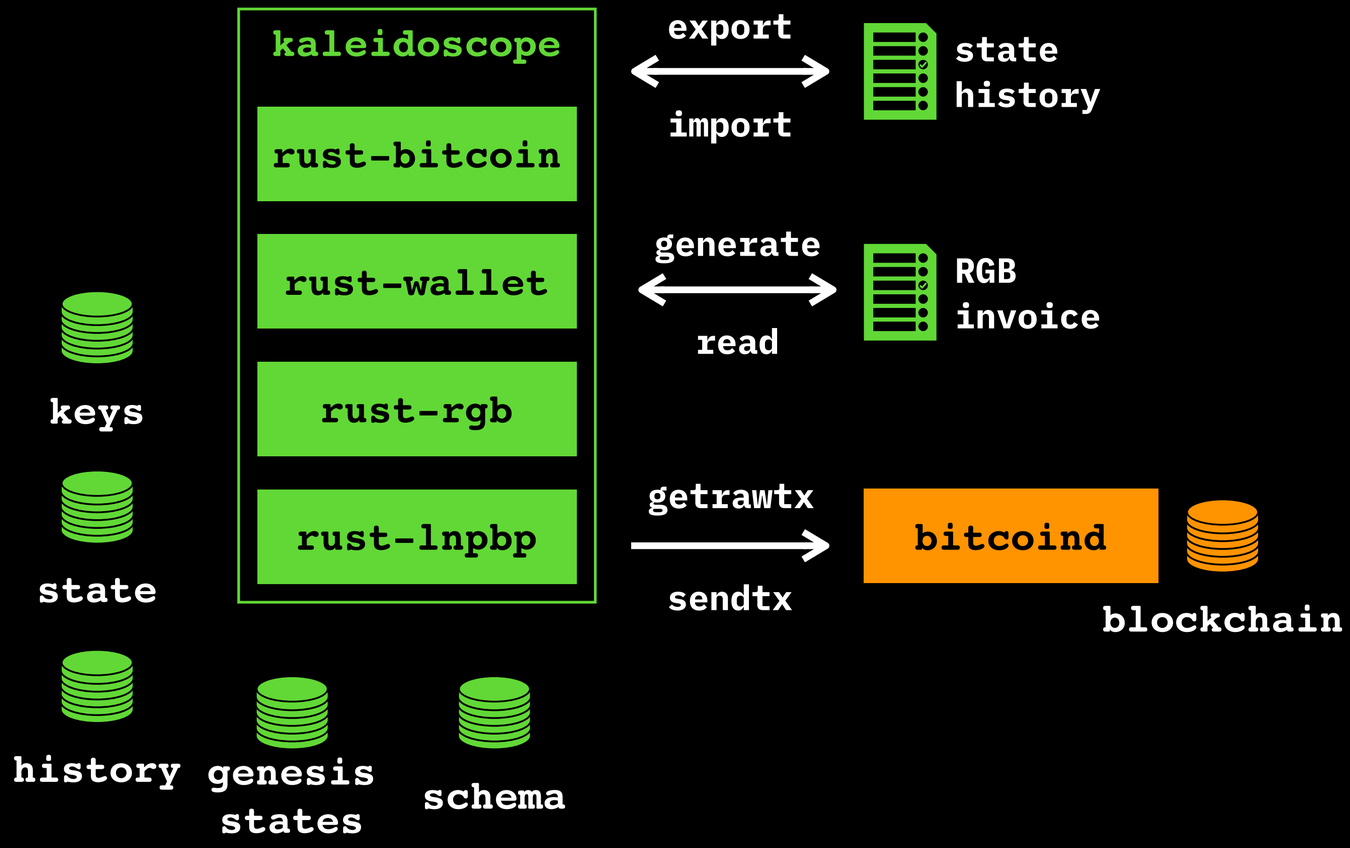
DevOps

- LNP/BP docker repository and images on Docker Hub:
  - GitHub repo: https://github.com/lnp-bp/docker
  - Docker Hub: https://hub.docker.com/u/lnpbp
- Now has
  - bitcoind mainnet
  - bitcoind signet (!)
- $ docker pull lnpbp/bitcoind:latest

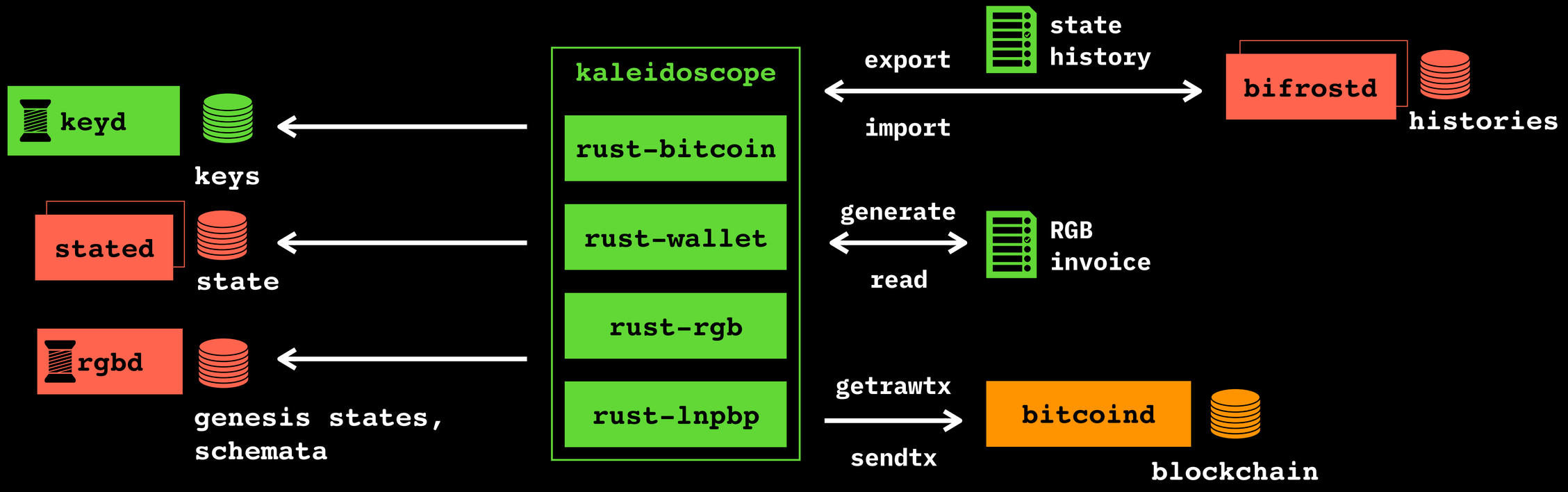
LNP/BP docker is a bitcoind with

- **-txindex:** requesting transactions for client-validated state
- **-without-wallet:** we do not need it
- **-without-gui:** we do not need it
- built on minimalistic latest Debian
- extensible with bitcoin.conf — and nothing else

Kaleidoscope: initial command-line wallet

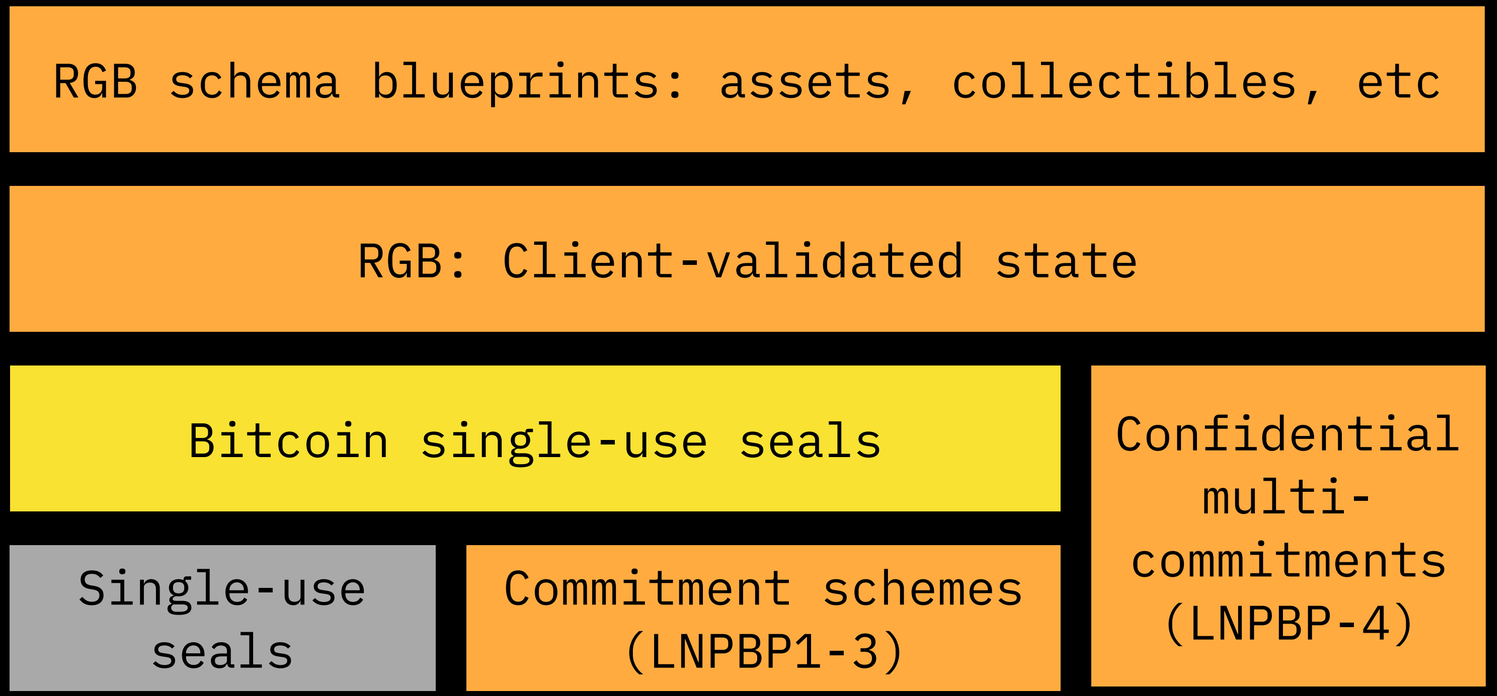


Kaleidoscope: command-line wallet stage 2

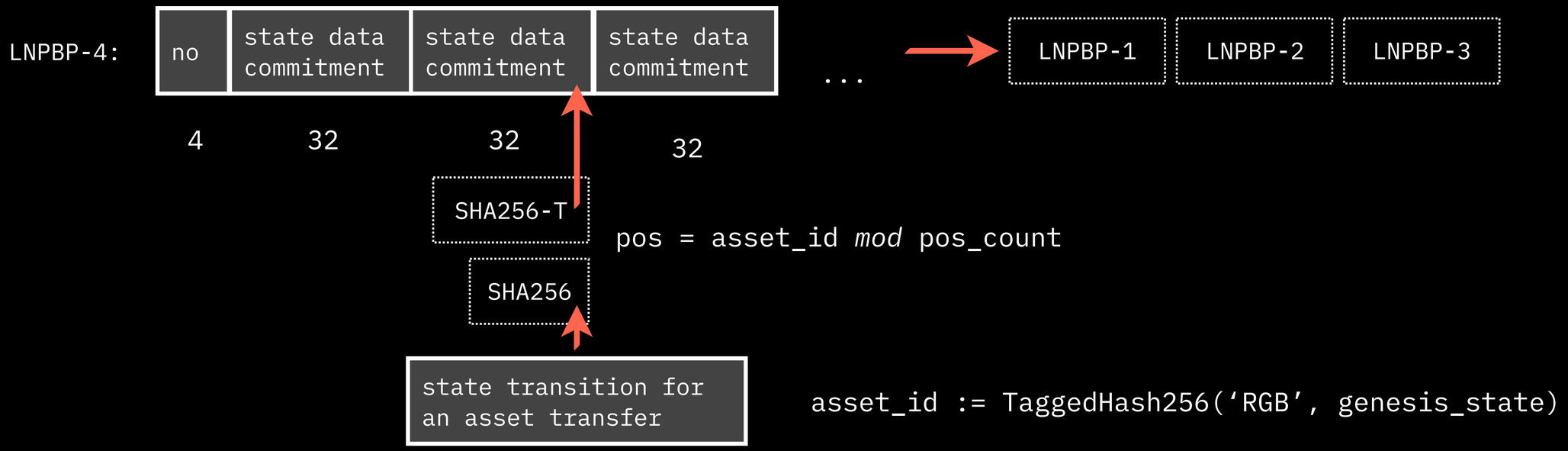


New developments for standards & code

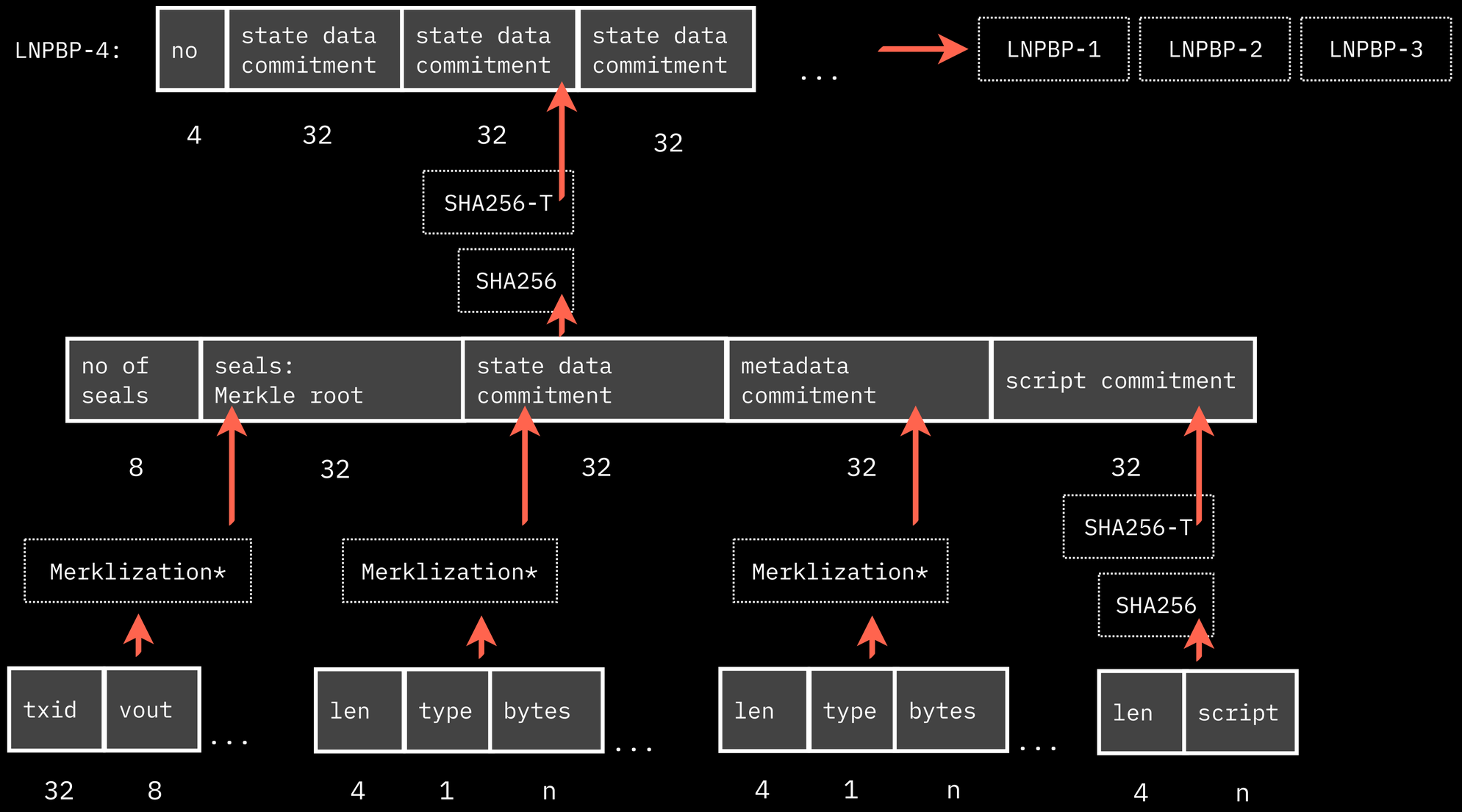
RGB Tech stack



LNPBP-4 with “Bloom filter” confidentiality



State transition commitment serialisation & process



```rust
mod runtime {
    pub struct Transition {
        pub commitment_source: serialize::commitment::Transition,
        pub known_seal_state: Vec<self::SealState>,
        pub meta: HashMap<self::MetaRecordId, self::MetaRecord>,
        pub script: Option<self::SimplicityScript>,
    }
}

mod commitment {
    pub struct Transition<const N: u32, const M: u32> {
        const seals_no: u32 = N,
        const fields_no: u32 = M,
        pub seals_cmt: MerkleRoot<N>,
        pub state_cmt: MerkleRoot<N>,
        pub metafields_cmt: MerkleRoot<M>,
        pub script_cmt: ScriptHash,
    }
}
```

Genesis state is different:

- Defines Schema
- Seals are not Merklized, must be validated by everyone
- Defines used Simplicity libraries
- May be signed

Sample: Fungible assets schema draft

```
name: Fungible assets
standard: LNPBP-6
schema_ver: 1
field_types:
  ver: u16
  schema: u256
  network: u8:0-6
  ticker: utf8:16 # Strings always have a fixed length specified after a colon;
                  # for shorter values the rest is extended with 0 bits
  title: utf8:64
  description: utf8:256
  url: utf8:1024
  fractional_bits: u8:0-256 # Integers always have a valid value bounds attached
  max_supply: amount # Amount is a special type, always equivalent to u256, plus requiring special validation rules
                     # (for any operation, sum of outputs must be equal to the sum of inputs, and must not overflow)
  dust_limit: amoun
seal_types:
  assets: amount
  issuance: none
  pruning: none
```

Sample: Fungible assets schema draft

```
proof_types:
  - name: primary_issue
    fields:
      ver: single
      schema: single
      network: single
      ticker: single
      title: single
      description: optional
      url: optional
      fractional_bits: single
      max_supply: optional
      dust_limit: single
    seals:
      assets: many
      issuance: optional
      pruning: single
    script: none
```

```
  - name: secondary_issue
    closes:
      issuance: single
    fields:
      description: optional
      url: optional
    seals:
      assets: many
      issuance: optional
      pruning: single
    script: none
```

```
  - name: history_prune
    closes:
      pruning: single
    fields: [ ]
    seals:
      assets: many
      pruning: single
    script: none


  - name: asset_transfer
    closes:
      assets: many
    fields:
    seals:
      assets: many
    script: none
```

Sample: Genesis state for a fungible asset issue

```
type_name: primary_issue
fields:
  ver: 1
  schema: sm1p9au5tw58z34aejm6hcjn5fnlvu2pdunq2vux5ymzks33yffrazxskfnvz5
  network: bitcoin:testnet
  ticker: PLS
  title: Private Company Ltd Shares
  description: Sample asset of no value
  fractional_bits: 0
  dust_limit: 1
  max_supply: 100_000_000
seals:
  - type_name: assets
    outpoint: 5700bdccfc6209a5460dc124403eed6c3f5ba58da0123b392ab0b1fa23306f27:0
    amount: 1_000_000
  - type_name: issuance
    outpoint: 5700bdccfc6209a5460dc124403eed6c3f5ba58da0123b392ab0b1fa23306f27:1
  - type_name: pruning
    outpoint: 5700bdccfc6209a5460dc124403eed6c3f5ba58da0123b392ab0b1fa23306f27:2
```

State transition validation

	Witness transaction	TxOuts for seals	State transition	Schema	Global script	Local script
Metadata	Must commit to	not related	Must be validly serialised	Validates presence and number of fields	may be accessed for state validation	may be accessed for state validation
Defined seals		Must be unspent		Validates types of allowed seals	may be accessed for state validation	may be accessed for state validation
Bound state		Defines ownership		Validates types of the defined state	Defines additional validity constraints	Defines additional validity constraints
Transition-specific script		not related		Must be allowed at the schema level		