RGB: Private and scalable smart contracts for Bitcoin and Lightning Network

RGB

RGB history

- Originally proposed by **Giacomo Zucco** & **Peter Todd** in 2016
- Engineered and developed by **Dr. Maxim Orlovsky**
- Supported by **Tether Inc/Bitfinex, Pandora Core AG** & other sponsors in early 2019
- **Pandora Core AG** is the leading technological force behind the development
- Governed by non-profit **LNP/BP Standards Association**, Switzerland

RGB and beyond

- RGB: **non-blockchain** (but client-side-validated) smart contracts
  - Much more _privacy_ (even more than blockchain-based ZK)
  - Much more _scalability_ (works over P2P non-consensus networks like lightning)
  - Much more _safe_ programmability due to separation of concerns
  - Much more _ownership_ instead of “governance”
- Pure **cypherpunk** stuff
  - Created by coders & scientists
  - No token!
  - 100% non-profit, open-source, but still anarcho-capitalistic (strong ownership focus)
  - Developed & maintained by Swiss LNP/BP Association (non-profit), which we plan to grow into something alike Linux Foundation & IETF in the future for LNP/BP stack
  - “John Galt’s” solution to the world problems

What is RGB?

Client-validated state, bearer rights and smart contract system working at Layer 2/3 in Bitcoin and Lightning Network.

- Works with **Lightning Network**
- No on-chain usage nor trackable footprint: client-validated paradigm
- **Scales** independently from blockchain
- **Zero-knowledge** & privacy built on best research-based products
  - Mimblewhimble: Bulletproofs by Andrew Poelstra
  - Liquid: Confidential Assets by Blockstream

Smart contract is

- A pre-arranged *agreement*
- of trade (i.e. mutual voluntary exchange of *goods*)
- automatically executed under certain *conditions*, where “automatic” means
  - anonymous: no KYC is done
  - trustless: no need to do KYC to protect from the failure to execute contract

Smart contract components

- Agreement: the code
- Goods: digital assets
- Conditions: contract parties or external actors able to call some code

Ownership & access: core properties

- **Ownership**: digital assets must be owned by a well-defined party
- **Access**: only well-defined parties should be able to call the contract execution

Pure blockchain/layer 1 approach is wrong:

- Mixing ***code***, ***ownership*** and ***access rights*** into a single layer (“blockchain”)
- which is inherently ***unscalable*** and well-trackable (***anti-privacy***) since VERIFICATION is needed by the whole world
- With Turing-complete ***code*** operating at the same level, ***compromising security***
- Running ***non-censorship-resistant*** consensus algorithms (PoS, PoW forks with small hashing power)

Challenges with digital assets today

- **Low scalability**
  - limited by blockchains, which are inherently unscalable
  - no layer 2 assets
- **Poor privacy**
  - Everyone in the world sees the transactions
  - Zero-knowledge is nearly absent for the assets even if it is present in blockchain (Monero, Grin, Beam, …)

Challenges with digital assets today

- **Inefficient smart contracts**
  - Asset ownership is mixed with contract business logic
  - Pseudo-decentralized (governance problem)
  - Not formally verified languages (security problem)

RGB was created to solve these issues

RGB is:

- “Sharding made right”
- “DAG made right”
- “Digital assets made right”
- “Smart contracts made right”
- “Confidentiality made right”

What is possible to do with RGB?

- Fungible assets & securities
  - Centrally or federation-issued
  - Issued anonymously or publicly
  - With possible secondary issuance, demurrage, inflation,
- Different forms of bearer rights
- Non-fungible assets (collectibles, game skins, art tokenization)
- Decentralized digital identity & roaming profiles
- Complex accounting systems & utility tokens

And it’s all:

- Scalable
- Confidential
- Working over Lighting Network
- With DEX functionality
- Operating as a bearer instrument

Examples of RGB Smart Contracts

- Each asset (fungible or non-fungible) issued by somebody is a separate smart contract
- Each root identity (like “master identity key”) is a separate smart contract
- Each case of provable audit log on RGB (like a patient’s medical history at a hospital) is a separate smart contract

RGB Smart Contract consists of

- Single smart contract **Genesis** node
  - Created by issuer
  - Committing to Schema
  - No commitments in bitcoin transaction graph
- Branching tree of **State transition** nodes
  - Created by owners during state ownership transfers
  - Always committed into transaction graph
  - Linked to each other (up to genesis) with single-use-seals
- **Simplicity scripts**, taken from Schema, Genesis and direct upstream line of transitions
  - Extend each other according to parent node rules
    (i.e. Genesis can add scripts only when allowed by Schema, state transition — only when allowed by Schema AND Genesis AND all previous state transitions)

Schema

- Shared by many contracts, i.e. sort of a “**contract type**”
- Defines rules for client-side validation of smart contract nodes (i.e. genesis and state transition) at both per-node level and as an upstream DAG
- Wallets, exchanges, payment providers etc integrate RGB schemata, not particular smart contracts (for instance, they integrate Fungible Assets Schemas, not USDT or particular asset)
- Immutable for eternity by social consensus
- Not committed to bitcoin blockchain (b/c no reason to do so)
- Smart contract is created under certain schema when it includes a hash of the corresponding schema data+structure+scripts

Initial list of Schemas

- Can be vendor-defined
- To have a broad support by software, most common must be standardized

Subject	Schema name	LNP/BP Standard	Analog to
Fungible assets	RGB-20	LNPBP-20	ERC-20
Collectibles	RGB-21	LNPBP-21	ERC-721
Reputation/indentity	RGB-22	LNPBP-22	n/a
Audit log	RGB-23	LNPBP-23	n/a

Smart contracts

	"Ethereum-style"	RGB
• **Parties of the agreement**	loosely defined	issuer and current owners
• **Agreement:**	Bockchain-stored contract + ABI file	Client-stored contract genesis + state transitions
- **Current state**	blockchain-stored data: ★ publicly visible ★ non-confidential ★ non scalable ★ no 2nd layer support	client-stored data: ★ no chain analysis ★ confidential ★ scalable ★ 2nd layer support
- **State change rules**	custom EVM code	schema & simplicity script
- **Ownership rights**		bitcoin script
• **Mutability**	Pseudo-immutable: immutable in promice, censored my miners & creators in fact	Well-defined mutability rights at genesis & schema level by issuer Mutable by new owners within the scope of rules

Problem 1: Blockchain does not scale

Problem 2: Blockchain is transparent

1. There always must be an owner

- Smart contract state is not a “public good” (Ethereum/“blockchain” approach);
  it must always have a well-defined ownership (private, multisig…).

- RGB defines ownership by binding/assigning state to Bitcoin transaction outputs with single-use seals: whoever controls the output owns the associated state

- I.e. RGB leverages Bitcoin script security model and all its technologies (Schnorr/Taproot etc).

2. State ownership != state validation

- Ownership defines WHO can change the state
- Validation rules (client-side validation) define HOW it may change

2. State ownership != state validation

- Ownership controlled by Bitcoin script, at Bitcoin blockchain level (non-Turing complete)
- Validation rules controlled by RGB Schema with Simplicity script (Turing-complete)

This allows to avoid mistake done by “blockchain smart contracts” (Ethereum/EOS/Polkadot etc): mixing of layers & Turing completeness into non-scalable blockchain layer

Also it makes possible for smart contracts to operate on top of Layer 2 solutions (Lightning Network)

RGB Smart Contracts:

- Bitcoin script: bare, hashed and Taproot
  - Multisigs, state channels, swaps…
- Scriptless scripts: private, less footprint

ownership:

single-use-seals

- RGB
  - Using schema & simplicity language
  - No blockchain footprint
  - Confidential
  - Nearly fully Turing-complete

state validation

client-side validation

Thus, RGB smart contract is:

- Distributed system
- Where nobody has the complete view of the current state
- But it is still globally consistent (has consensus) because of:
  - Single-use seals based on bitcoin PoW
    (with possible LN as an intermediary)
  - Social consensus on the same client-side validation rules (Schema)
- Only owners has access to their owned state + a slice of state history DAG directly related to the owned state

Rights management under RGB smart contract

- RGB rights:
  Smart-contract defined types of actions, which can be taken only by a party owning some part of the smart contract state.
  - Ownership of the assets
  - Ownership of the identity
  - Right to inflate asset supply
  - Right to create child identities
  - Right to prune/prune assets
  - …

Rights management under RGB smart contract

- Types of allowed rights are **defined** at Schema.
  They named *state types*
  (b/c each right must have some current state/value, even if this is an "empty value")

- Initial rights are **assigned** by the contract issuer in Genesis

- Rights can be **transferred** (together with the new state value) to new owners with *state transitions*

- Rights (state) **ownership** is controlled with bitcoin scripts via *single-use-seal mechanism*

- Who will be the next owner for particular right is always defined by the party that currently have that right (i.e. *state owner*)

- *Client-side validation*: rights transfers MUST be always **validated** backwards by the new owners according to validation rules.

Right transfer/state transition validation rules

- **Defined by Schema** using two main instruments:
  - Schema structure (how rights can be decided among descendants)
  - Simplicity scripts (how the state of some rights may evolve).
    For instance, for assets script requires that a sum of outputs must be equal to a sum of outputs.
- Can be further **restricted** (and not extended) at the level of **genesis** and each state transitions
- **Validated by new owners** backwards up to the genesis within a particular subgraph
- Violation of the rights in one smart contract ownership branch does not affects smart contract integrity in other branch

Security measures

- Each right (i.e. state) does not have a direct access to the information on the state under other rights
- If required, rights can have a “shared state” using metadata; Schema and Genesis explicitly defines whether this is allowed
- State can be “hidden” (made confidential) with zero knowledge; which state MUST be hidden is defined by the Schema

Schema defines

- Types of metadata and their value restrictions
  (like max length for strings; max value for integers etc)

- Types of rights (i.e. state) and their value restrictions

- How rights transfers (state transitions) can be organized:
  - which metadata they must provide
  - which rights can be (or must be) transferred jointly
  - history validation rules for each of the rights, defined using Simplicity
    (like “sum of outputs must be equal to the sum of inputs)

- How these rules can be further limited - or extended - at the level of genesis and individual state transitions

Main components of RGB

- 1. Commitments in transactions, proving unique history
  - private
  - zero storage cost
  - work both with blockchain (layer 1) and Lightning Network (layer 2)
  - meaning extreme scalability

- 2. Off-chain data & code held by asset/contract owner
  - zero blockchain storage cost
  - assets are linked to transaction outputs, which define their ownership & prevent double-spending (single-use seals)
  - off-chain smart contract code defines asset evolution

RGB schema is not a script,
but is a **data structure**

may (or may not) contain script extensions
(for dynamic part of the validation)

RGB Schema: Contractum Language

github.com/rgb-org/contractum-lang

- Just a convenient way of defining new schema requiring no rust coding knowledge
- Impossible to define invalid/internally inconsistent schema
- Not a script language!
  This is data definition language, like HTML, XML, YAML etc
- May contain scripts for dynamic validation
  (once VM for RGB will be completed)

```
final contract RGB20 implements FungibleAsset {
    field ticker: str
    field name: str
    field contractText: str?
    field issuedSupply: value
    field precision: uint8
    field created: timestamp

    assigns inflationRight: value*
    assigns assetsOwnership: value+ {
        validate() {
            diff = parent[..].sum() - self[..].sum()
            if diff != 0 {
                error(failures.FungibleAsset.Inflation(diff))
            }
        }
    }
    assigns renominationRight: decl?
    assigns burnEpochOpening: decl?

    fn validateSupply() {
        if self.issuedSupply != self.assetsOwnership[..].sum() {
            error(failures.FungibleAsset.IncorrectIssue())
        }
    }

    validate() {
        validateSupply()
    }

    transition AssetTransfer fulfills assetsOwnership+ {
        assigns assetsOwnership: value+
    }
```

Which VM to use for RGB?

- **Simplicity:**
  - not VM :(
  - not ready :(
  - no toolchain for devs :(
  - way too complex to understand how to program
- **WASM:** (and actually the same applies to **LLVM**, **JVM**, **CLR** + they have even more negative sides)
  - requires a lot of customization for blockchain/client-side-validation applications
  - the only version is from Parity Labs, doing Polkadot, infamous with creating contracts multiple times hacked
- **IELE** or **kEVM**:
  - EVM on drugs :(
  - a lot of custom unneeded functionality for non-bitcoin type of cryptocurrency
- do something custom?

AluVM - from “arithmetic logic unit”

github.com/internet2-org/aluvm-spec

- Purely functional & arithmetical: each operation is an arithmetic function
- No external state; converts set of inputs into false/true validation result
- Extremely robust & deterministic: no exceptions are possible
  - no stack (register-based VM)
  - no random memory access
  - no I/O, memory allocations
- If it compiles, it will always run successfully
- Easy to be implemented in hardware, like in FPGAs

RGB vs existing alternatives

RGB compared to Liquid Confidential Assets:

- Works with Lightning Network
- Replaces Large Borromean signatures range proofs with modern Bulletproofs
- No blockchain space consumption!
- Universal smart contract system
- Works on Bitcoin mainnet, does not require federation

RGB compared to OMNI/Colored coins/ Counterparty:

- No blockchain consumption
- Much higher privacy
- Works with LN without its modifications

RGB compared to Ethereum/EOS/other "corporate blockchains”:

- RGB is *NOT* a blockchain!
- Works on and with Bitcoin: the only censorship-resistant unconfiscatable hard money

Omni BOLT compared to RGB LN:

- Breaks BOLT message compatibility
- Breaks BOLT tx structure compatibility
- No backports from LND
- No TLV extensions
- Requires separate nodes for OMNI Bolt and Bitcoin LN
- Requires Omni Core backend, can't work with just Bitcoin Core

RGB & other Bitcoin tech

- Interoperable with Liquid
- Does not require changes to LN layer
- Leverages Taproot & Schnorr on the base layer
- Unified invoicing for RGB, Bitcoin and LN
- Can work with existing node implementations, but also has Rust-implementations (BP Node, LNP Node, RGB Node)

What does RGB have now?

- https://www.rgbfaq.com/community/getting-familiar-with-rgb

RGB and beyond

- RGB: **non-blockchain** (but client-side-validated) smart contracts
  - Much more _privacy_ (even more than blockchain-based ZK)
  - Much more _scalability_ (works over P2P non-consensus networks like lightning)
  - Much more _safe_ programmability due to separation of concerns
  - Much more _ownership_ instead of “governance”
- Pure **cypherpunk** stuff
  - Created by coders & scientists
  - No token!
  - 100% non-profit, open-source, but still anarcho-capitalistic (strong ownership focus)
  - Developed & maintained by Swiss LNP/BP Association (non-profit), which we plan to grow into something alike Linux Foundation & IETF in the future for LNP/BP stack
  - “John Galt’s” solution to the world problems

RGB vs Ethereum recap

	"Ethereum-style"	RGB
• **Parties of the agreement**	loosely defined	issuer and current owners, good role distinction
• **Agreement:**	Bockchain-stored contract + who-knows-who-keeps ABI file	Client-stored data only
- **Current state**	blockchain-stored data: ★ publicly visible ★ non-confidential ★ non scalable ★ no 2nd layer support	client-stored data: ★ no chain analysis ★ confidential ★ scalable ★ 2nd layer support
- **State change rules**	custom EVM code	schema & simplicity script
- **Ownership rights**		bitcoin script
• **Mutability**	Pseudo-immutable: immutable in promise, *de facto* censored my miners, Vitalik® & contract creators	Well-defined mutability rights at genesis & schema level by issuer Mutable by new owners within the scope of rules

RGB vs Ethereum in simple words

Ethereum

is a mess:

- needless token (why do we need ETH?)
- no clear ownership rights at any level
- governance worse than with a government
- all layers mixed together
  - bugs & hacks
  - unscalable
  - low privacy
- constant hard forks
- contract can contain backdoors

RGB

quite the opposite:

- no token
- bitcoin-level safety guarantees of ownership
- scalable over layer 2 *and 3* solutions
  (LN, DEX, smart contracts on top of RGB)
- extreme confidentiality
- clear ownership rights & mutability
  due to client-side-validation
  - no miners involved
  - issuers lose control the moment they create a contract
  - owners are always in control and know all terms & conditions upfront; no backdoors are possible

“Multi-blockchain world” criticism

- Only a single blockchain should serve censorship resistance needs; other blockchains are not needed since they are either non-confidential or unscalable and insecure
- All data must be kept by data owners (client-side-validation); they may pay for delegating that, but not in “communistic way” (like with blockchains)
- What we need is isolated contracts (like with RGB), interoperable with each other via layer 2 & 3 (LN) – instead of connected “internet of blockchains”
- True proof of stake is a stake you can lose for breaking RGB contract in a multipeer LN channel – and not public & censorable blockchain shit tokens

RGB mission

- Much more _privacy_ (even more than blockchain-based ZK)
- Much more _scalability_
  (it works over P2P non-consensus networks like lightning)
- Much more _safe_ programmability due to separation of concerns
- Much more _ownership_ instead of “governance”

What will drive RGB adoption?

- Scalability

Governments like blockchains because they can control validators and do chain analysis (even more with new “ZK”-blockchains)

Enterprise follows the government because it’s scared of its enforcement power, so enterprise/corporates comply/cooperate (like they did even with Nazis)

BUT: *how the fuck they are going to scale with that shit?*
Blockchains, fortunately for us, do not scale!

So, our chance is: before the governments will understand what the new shift of paradigm on client-side-validation coming from cypherpunks is about (and previously this took years for bitcoin), normal companies will have a window to start using RGB because … they need digital scalability!

Building adoption

- User- and dev-facing tools are critical!
- Tools may be for-profit/commercial, since we need to provide support for businesses. Finally, we are anarcho-capitalists, not socialists!
- Open-source and “free to use if self-supported, paid for professional support” set of tools is required for real adoption
- Devs of the original protocol are able to deliver initial toolset to the market since they understand all possibilities of the new protocols
- Pandora Core AG run by RGB devs gives initial set of tools for devs & users matching those criteria

Build on LNP/BP Association tools

- These tools are just integration of & UI on top of LNP/BP tools, simplifying life for users & devs:
  - Client-side-validation and LNP/BP Core Libraries
  - RGB Core library & Node
  - LNP Core library & Node
  - Descriptor wallet library
- You can DIY your tools, even commercial
- Even more: you can use Pandora Core tools for free to start using RGB tomorrow!

What are these tools?

- **Bitcoin Pro:** MIT-licensed tool for professional asset issuers (and even non-RGB professional bitcoiners)
  GTK+-based, all desktop platforms, pure Rust

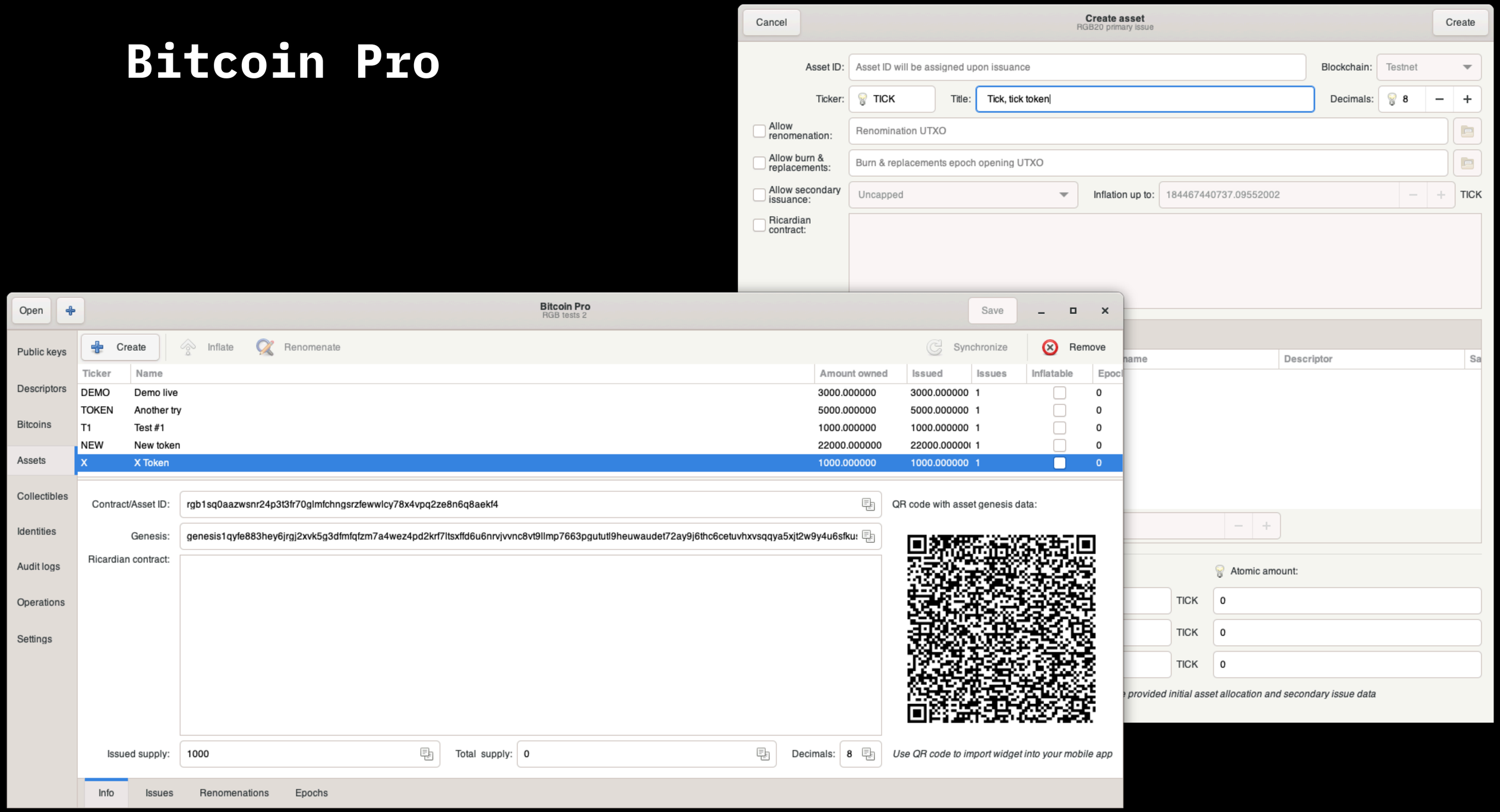
- **Citadel SDK**: MIT-licensed SDK for wallet devs to get RGB & LN running spending just a week on integration

- **MyCitadel** suite of products:
  wallet apps, appliances, private cloud

- …one more thing which we will uncover today

- …even more LN & DEX-related things to come by the end of the year

Bitcoin Pro


Cancel
Create asset
RGB20 primary issue
Create
Asset ID:
Asset ID will be assigned upon issuance
Blockchain:
Testnet
Ticker:
TICK
Title:
Tick, tick token
Decimals:
8
Allow renomenation:
Renomination UTXO
Allow burn & replacements:
Burn & replacements epoch opening UTXO
Allow secondary issuance:
Uncapped
Inflation up to:
184467440737.09552002
TICK
Ricardian contract:
Open
Bitcoin Pro
RGB tests 2
Save
Public keys
Descriptors
Bitcoins
Assets
Collectibles
Identities
Audit logs
Operations
Settings
Create
Inflate
Renomenate
Synchronize
Remove
Ticker
Name
Amount owned
Issued
Issues
Inflatable
DEMO Demo live 3000.000000 3000.000000 1 0
TOKEN Another try 5000.000000 5000.000000 1 0
T1 Test #1 1000.000000 1000.000000 1 0
NEW New token 22000.000000 22000.000000 1 0
X X Token 1000.000000 1000.000000 1 0
Contract/Asset ID:
rgb1sq0aazwsnr24p3t3fr70glmfchngsrzfewwlcy78x4vpq2ze8n6q8aekf4
Genesis:
genesis1qyfe883hey6jrgj2xvk5g3dfmfqfzm7a4wez4pd2krf7ltsxffd6u6nrvjvvnc8vt9llmp7663pgututl9heuwaudet72ay9j6thc6cetuvhxvsqqya5xjt2w9y4u6sfku
Ricardian contract:
QR code with asset genesis data:
Issued supply:
1000
Total supply:
0
Decimals:
8
Use QR code to import widget into your mobile app
Info
Issues
Renomenations
Epochs
Descriptor
Atomic amount:
TICK
0
provided initial asset allocation and secondary issue data

4:24 PM Thu Mar 18

100%

My Citadel

Default

X Token

1,000.000000 X

Unknown

Bitcoin (testnet)

0.000100 tBTC

Proof of Work

Send

History Balance

Invoice Accept

Balance

ADDRESS

TB1zswt4lsmevzfqyvldq8rp3vjadewq6wet**t48d5c**

Balance: 0.0001 tBTC

3680829e9e50cc9a75d01ce855c7701c4b830570a66610db564ce31247ef77eb 0.000100 tBTC

Output number: 0

4:24

My Citadel

Default

Bitcoin (testnet)

0.000000 tBTC

Proof of Work

INVOICE | PAY | ACCEPT

HISTORY | BALANCE

2021-03-18T14:07:28 1,000.000000 tBTC

4:25

Dissmiss

Contract details

CONTRACT NAME

Default

GENERAL INFO

Contract id	id1uvpj5xjxmcl2lqa9hpsa7vn47ec4v8ntk5m6rgzqma89qtntq5usftt52w
Contract type	Current account
Address format	P2WPKH
Alternative format	P2WPKH-in-P2SH
Total addresses	2147483648
Network	Bitcoin testnet
RGB compatible	yes

Used addresses

DESCRIPTOR

Type	Wpkh

4:25

Address list

Address details

Wallet	Default

Address

tb1qg0a5exfj7wt0j4h6u85zwj4z2uydzzzqrn6mju

Derivation index	1

BITCOINS

Bitcoin (testnet)	**0.000000** tBTC
Testnet satoshis	**0** tSats.
Currency equivalent	**0.000000** USD

RGB20 ASSETS

No assets

UNSPENT TRANSACTION OUTPUTS (UTXOS)

No unspent outputs

Citadel SDK

- **Single point of integration** for all cool stuff
  (RGB, LN, Taproot, miniscript, multisigs & descriptors; DLC & DEX in the future …)

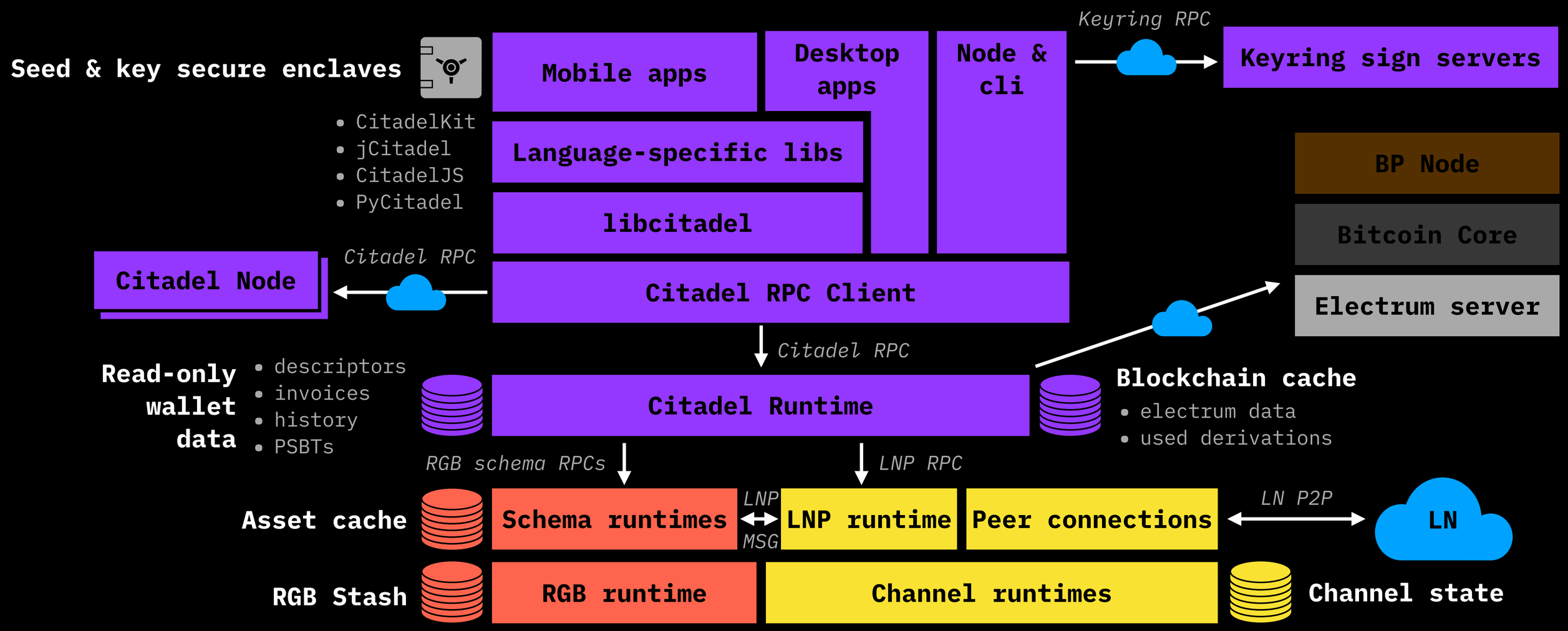
- **Simple API** hiding complexity of RGB and LN nodes management,
  Electrum integration & and wallet data storage
  (call just one method for RGB or LN payment with an invoice)

- **Native libraries** with OOP API for:
  Apple platforms, Android & Java[WIP], NodeJS[WIP], Python[WIP]

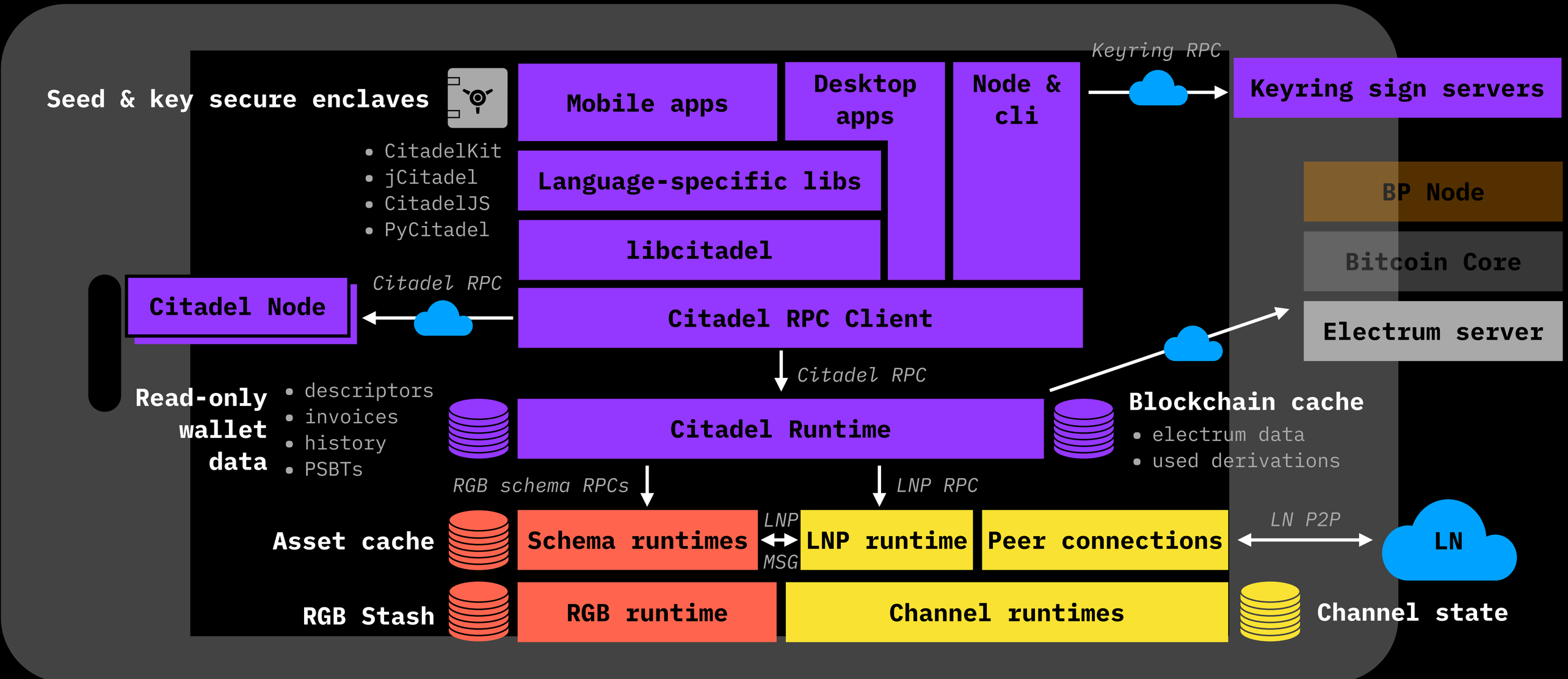
- **MIT-licensed** with tech **support** & integration services by Pandora Core AG

- Can be used in **personal/enterprise** setups with shared wallets
  (across devices or company employees) using
  self-hosted **MyCitadel Box** or **private cloud-hosted MyCitadel Node** by Pandora Core AG
  with revenue sharing for wallet development companies

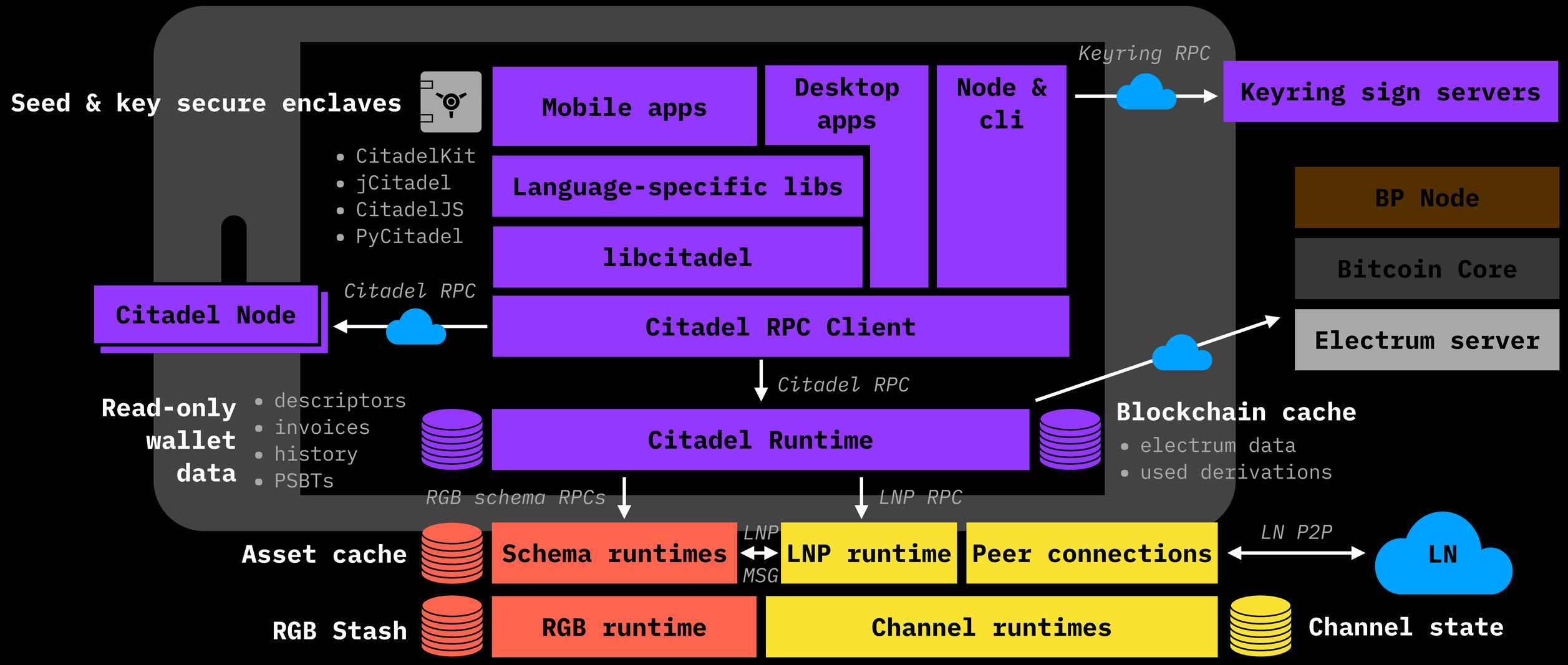
Wallet architecture (based on Citadel SDK)



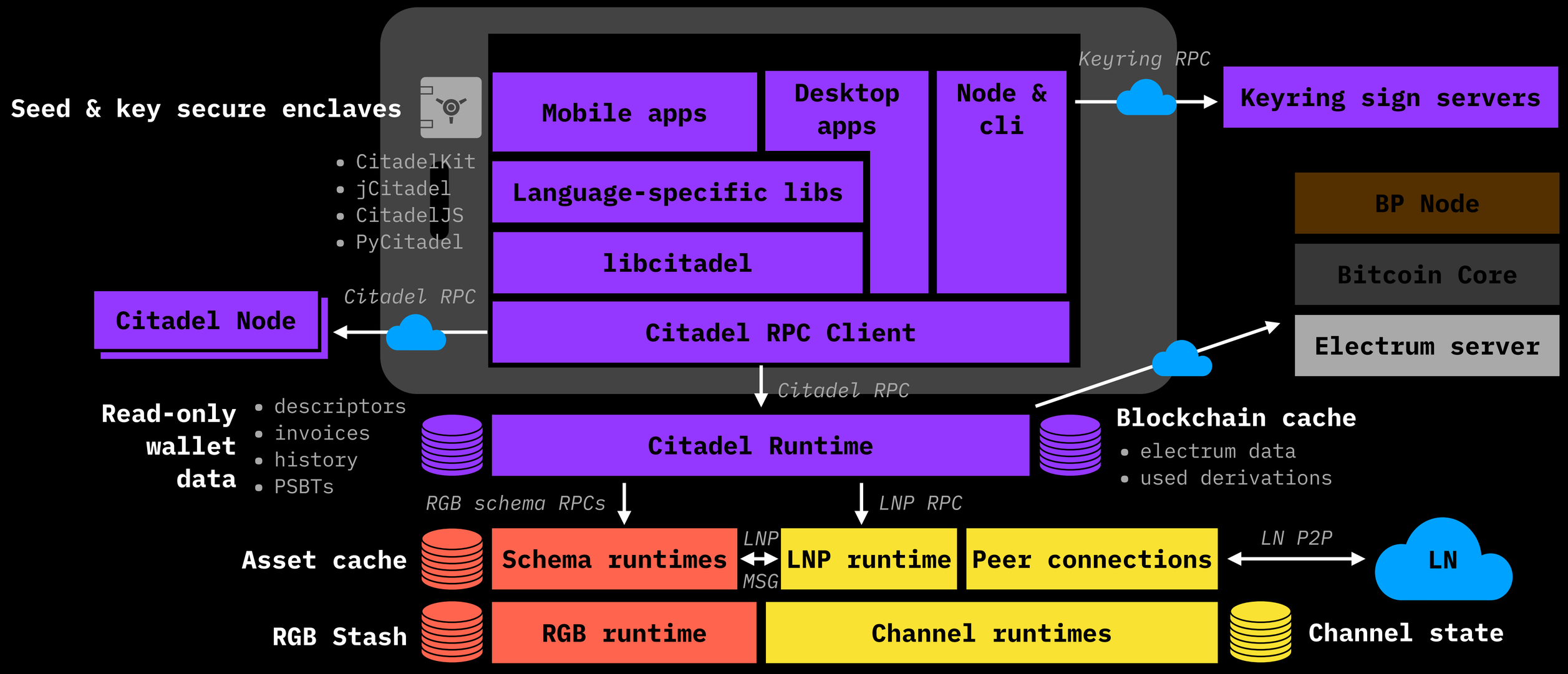
Everything can run on mobile!



…or with external RGB & LN nodes



…or just as a client



When Citadel?

- **Citadel SDK** technology preview available today on github.com/MyCitadel
  - main *Citadel runtime* implementing 100% of business logic for RGB, miniscript & multisigs (LN, Taproot expected by June)
  - C- and Swift class library (*libcitadel* & *CitadelKit*) NodeJS, Java & Python are expected by June and end of summer
  - Wallet developers technical support starts mid-summer (upon LN completion)
- **MyCitadel Node** for self-hosting or enterprise setups at v0.1 beta
- **MyCitadel Box** – hardware appliance running MyCitadel Node, in cooperation with Nodl – expected by the end of summer
- **MyCitadel Cloud** (in cooperation with Nodl) – this autumn

Keyring: signature server infrastructure

- Manage **multisig policies** for a hot custody
- …including **enterprise** + **personal/family setups**
- 100% **PSBT** compatible, **Taproot** & MuSig-ready
- **RGB** compatible (and supports pay-to-contract key tweaks)
- **Part of Citadel suite**: works well with the Citadel SDK and MyCitadel wallet, hardware & private cloud setups
- Under development: expected **by the end of year**

RGBex.io

The first RGB explorer

… and it is not a “blockchain explorer”: no chain analysis, tracking etc.
Just publishing & sharing information about your assets

RGBex is

- Playground for RGB assets, including NFTs
- Playground for Bifrost infrastructure experiments
- Playground for RGB-based identity management

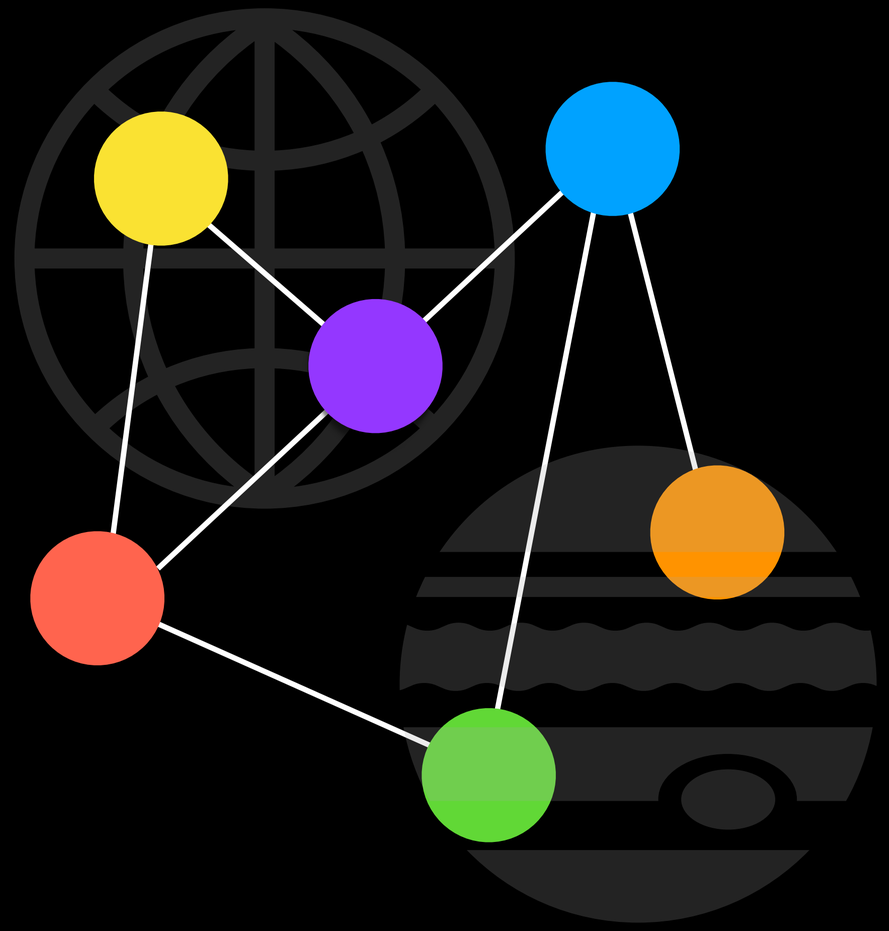
Architecture: Internet2 by LNP/BP Association

- Everything made of modular compact **microservices**:
  you can distribute your software across **mobile devices & servers**,
  including **cloud**, **Docker**, Kubernets the way you like

- Talking to each other over binary encrypted LN-style protocol
  on top of **ZeroMQ**: no old, slow and insecure JSON RPCs etc,
  no HTTP or plain text data

- All business logic is written in **Rust**:
  - Blazingly-fast
  - Deterministically secure and highly robust in runtime

- API is wrapped in language-specific **class libraries with good OOP abstractions**



Internet2

github.com/internet2-org

We are hiring

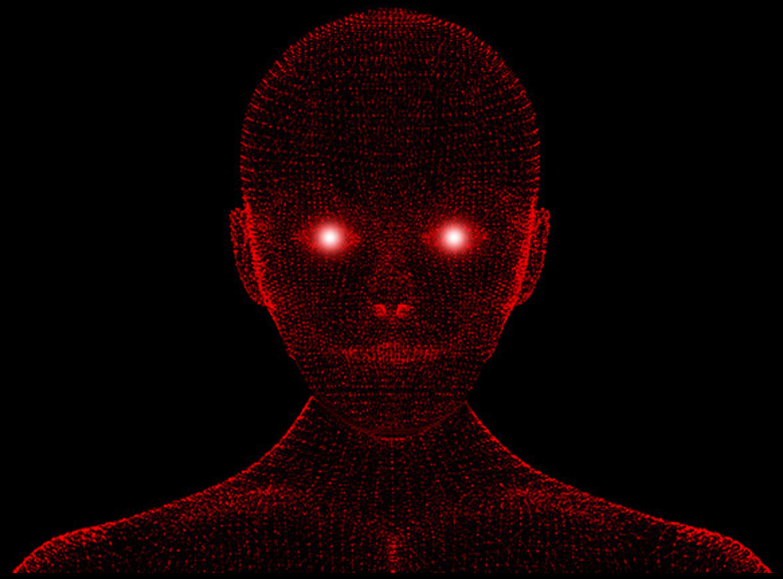
- Cool **rust** devs: **LNP/BP Association**
  for working on *LNP Node*, *NFTs*, *identity*, *LN DEX*, *BP Node*
  Taproot, PSBT2, descriptors & miniscript
  Internet2 protocols
  work with Blockstream, Square Crypto engineers, TLS creator

- **Android dev**: **MyCitadel** wallet
  with Kotlin & Jetpack Compose experience

We are fundraising

- **LNP/BP Association** donations for 2021
  - private
  - corporate
  - LNP/BP membership
  - LNP/BP memorable tokens (zero-value indeed, just a memento :)
  - will have a separate presentation for 2021 Roadmap
  - Bitfinex/Tether Inc, Fulgur Ventures and Pandora Core were major contributors in 2020 & Q1 2021, but more scale & resilience is desired for the rest of 2021
- **Pandora Core AG** investments for further product development, marketing & support:
  - Citadel & MyCitadel suits (SDK, node, wallets, appliance, cloud, custody)
  - Bitcoin Pro enterprise
  - future (not yet) uncovered DEX-related products (joint project with **HodlHodl**)
- **Federation** participation to run RGB-wrapped decentralised-issued non-custodial* Bitcoin (RGB30)
  - Exchanges, wallets & cool tech guys
  - More programmability & privacy to Bitcoin!

Contact Olga Ukolova for details (Telegram: @dr_ukolova, E-mail: ukolova@lnp-bp.org or ukolova@pandoracore.com)



Thank you!

Olga ‘Dolores’ Ukolova, MD

ukolova@lnp-bp.org

- Co-founder and COO at Pandora Core
- Co-author of The #FreeAI Manifesto
- Twitter @OlUkolova