How RGB Colors Coins *in Secret*

w/ Hunter Beast

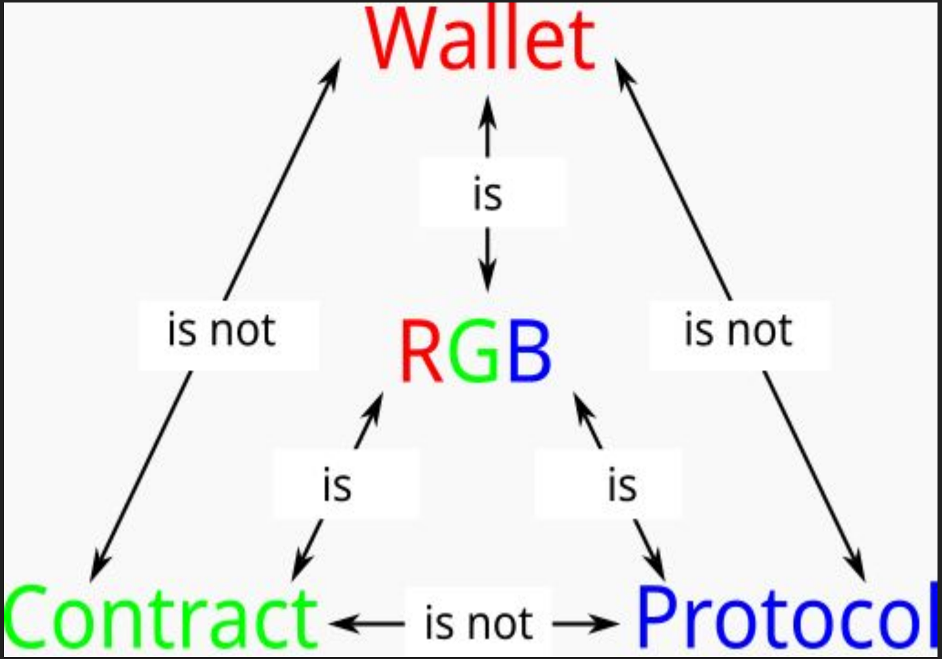
- Director of Engineering, DIBA
- DIBA - Digital Bitcoin Assets
- Bitcoin, Rust, RGB dev
- BitMask wallet - bitmask.app

What were Colored Coins?

- Alternate units of account
- OmniLayer - OG Tether issuance
    - Recently officially discontinued by Tether in favor of RGB
- According to Jimmy Song, OG colored coins failed due to:
    - No standardization, vs ERC-20 standard
    - No marketing, vs Ethereum Foundation marketing budget
    - Market wasn't ready, vs ICO niche
- Why?
    - Everything over 21M

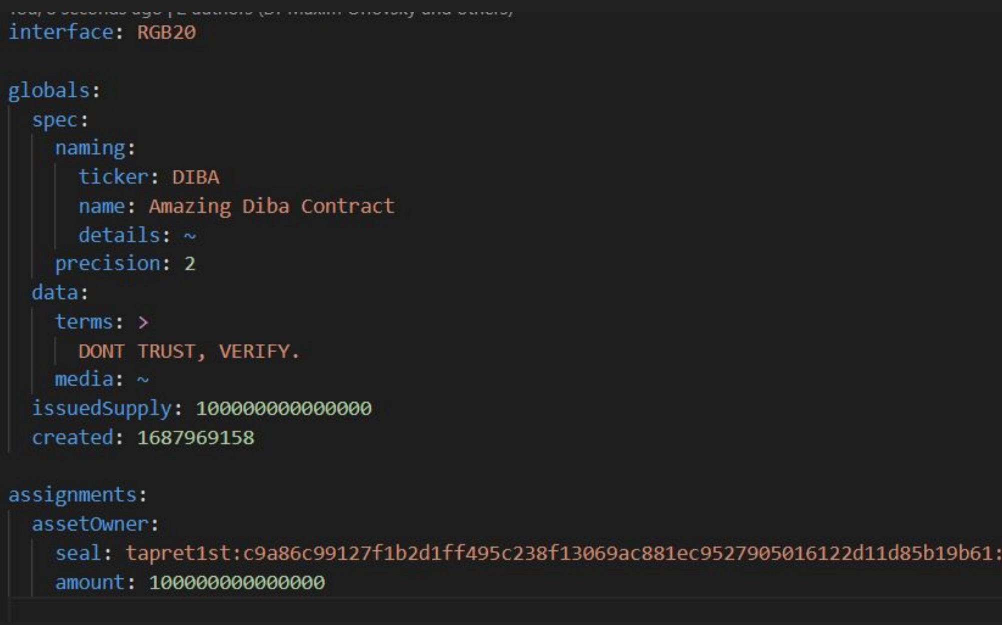
What is RGB?

- Generalized smart contract protocol
- Keeps contract data off-chain
- Contract execution occurs in clients
- Contract state is sharded amongst contract owners
- Strong privacy characteristics
- Secured by Bitcoin
- *Bitcoin-only, no other blockchain needed, other fee token than sats*



RGB-20

- Token metadata is specified (name, supply, precision) <- "global state"
- Tokens are minted against a UTXO <- "owned state"
- A UTXO proves three things:
  - This state has an owner ("Separation of ownership and state")
  - This state is currently valid ("Single-use seal")
  - Not double-spent (helps w/sharding)



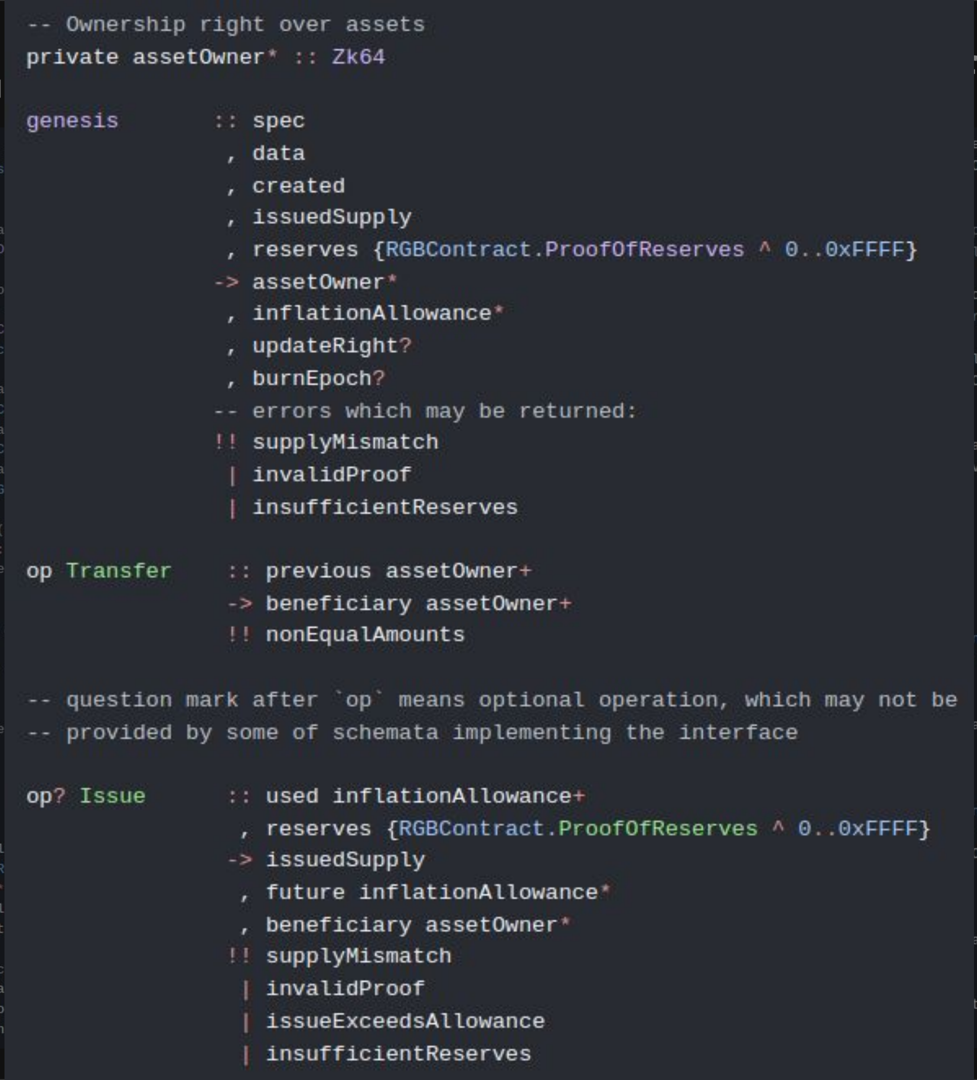
```
interface: RGB20

globals:
  spec:
    naming:
      ticker: DIBA
      name: Amazing Diba Contract
      details: ~
    precision: 2
  data:
    terms: >
      DONT TRUST, VERIFY.
    media: ~
  issuedSupply: 100000000000000
  created: 1687969158

assignments:
  assetOwner:
    seal: tapret1st:c9a86c99127f1b2d1ff495c238f13069ac881ec9527905016122d11d85b19b61:
    amount: 100000000000000
```

RGB20 Contractum Interface Specification

```
-- Defined by LNPBP-31 standard in `RGBContract.sty` file
import urn:ubideco:stl:6vbr9ZrtsD9aBjo5qRQ36QEZPVucqvRRjKCPqE8yPeJr#choice-little-boxer as RGBContract

interface RGB20
    -- Asset specification containing ticker, name, precision etc.
    global spec :: RGBContract.DivisibleAssetSpec

    -- Contract data and creation date is separated from the spec since it must
    -- not be changeable by the issuer.
    global data :: RGBContract.ContractData
    global created :: RGBContract.Timestamp

    -- State which accumulates amounts issued
    global issuedSupply+ :: RGBContract.Amount
    -- State which accumulates amounts burned
    global burnedSupply* :: RGBContract.Amount
    -- State which accumulates amounts burned and then replaced
    global replacedSupply* :: RGBContract.Amount

    -- Right to do a secondary (post-genesis) issue
    public inflationAllowance* :: Zk64
    -- Right to update asset Specification
    public updateRight?

    -- Right to open a new burn & replace epoch
    public burnEpoch?
    -- Right to burn or replace existing assets under some epoch
    public burnRight*

    -- Ownership right over assets
    private assetOwner* :: Zk64

    genesis       :: spec
                   , data
                   , created
                   , issuedSupply
                   , reserves {RGBContract.ProofOfReserves ^ 0..0xFFFF}
                  -> assetOwner*
                   , inflationAllowance*
                   , updateRight?
                   , burnEpoch?
                  -- errors which may be returned:
                  !! supplyMismatch
                   | invalidProof
                   | insufficientReserves
```

```
op Transfer     :: previous assetOwner+
                -> beneficiary assetOwner+
                !! nonEqualAmounts

-- question mark after `op` means optional operation, which may not be
-- provided by some of schemata implementing the interface

op? Issue       :: used inflationAllowance+
                 , reserves {RGBContract.ProofOfReserves ^ 0..0xFFFF}
                -> issuedSupply
                 , future inflationAllowance*
                 , beneficiary assetOwner*
                !! supplyMismatch
                 | invalidProof
                 | issueExceedsAllowance
                 | insufficientReserves

op? OpenEpoch   :: used burnEpoch
                -> next burnEpoch?
                 , burnRight

op? Burn        :: used burnRight
                 , burnedSupply
                 , burnProofs {RGBContract.ProofOfReserves ^ 0..0xFFFF}
                -> future burnRight?
                !! supplyMismatch
                 | invalidProof
                 | insufficientCoverage

op? Replace     :: used burnRight
                 , replacedSupply
                 , burnProofs {RGBContract.ProofOfReserves ^ 0..0xFFFF}
                -> future burnRight?
                 , beneficiary assetOwner+
                !! nonEqualAmounts
                 | supplyMismatch
                 | invalidProof
                 | insufficientCoverage

op? Rename      :: used updateRight
                -> future updateRight?
                 , new spec
```

RG ication

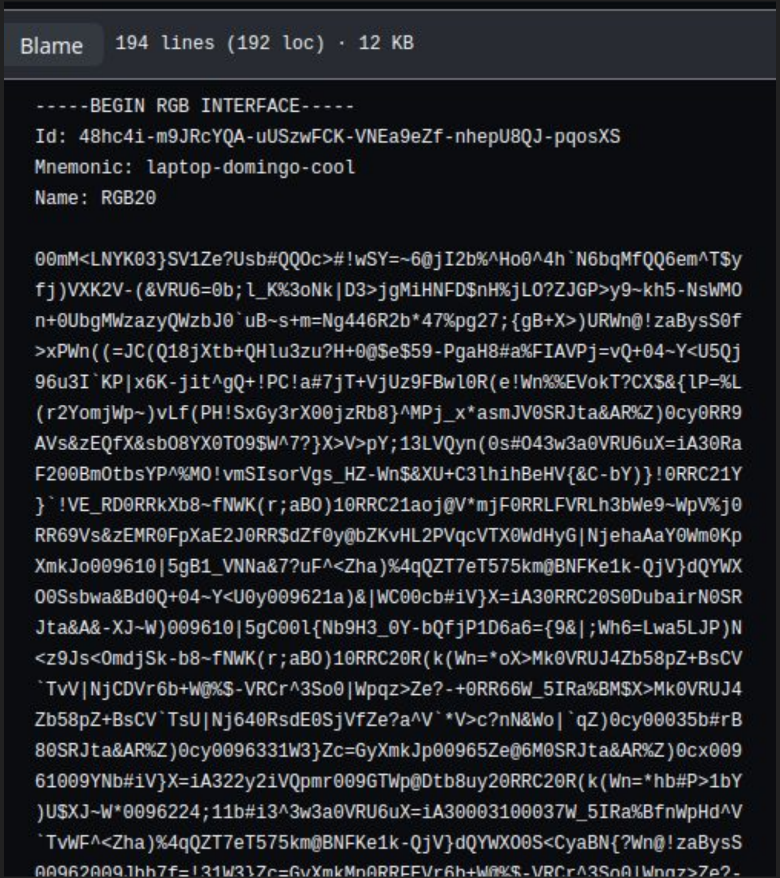


```
-- Ownership right over assets
private assetOwner* :: Zk64

genesis       :: spec
               , data
               , created
               , issuedSupply
               , reserves {RGBContract.ProofOfReserves ^ 0..0xFFFF}
              -> assetOwner*
               , inflationAllowance*
               , updateRight?
               , burnEpoch?
              -- errors which may be returned:
              !! supplyMismatch
               | invalidProof
               | insufficientReserves

op Transfer    :: previous assetOwner+
               -> beneficiary assetOwner+
               !! nonEqualAmounts

-- question mark after `op` means optional operation, which may not be
-- provided by some of schemata implementing the interface

op? Issue      :: used inflationAllowance+
                , reserves {RGBContract.ProofOfReserves ^ 0..0xFFFF}
               -> issuedSupply
                , future inflationAllowance*
                , beneficiary assetOwner*
               !! supplyMismatch
                | invalidProof
                | issueExceedsAllowance
                | insufficientReserves
```

```
-- Defined by LNPBP-31 standard
import urn:ubideco:stl:6vbr9Zrts

interface RGB20
    -- Asset specification conta
    global spec :: RGBContract.D

    -- Contract data and creatio
    -- not be changeable by the
    global data :: RGBContract.C
    global created :: RGBContrac

    -- State which accumulates a
    global issuedSupply+ :: RGBC
    -- State which accumulates a
    global burnedSupply* :: RGBC
    -- State which accumulates a
    global replacedSupply* :: RG

    -- Right to do a secondary (
    public inflationAllowance* :
    -- Right to update asset Spe
    public updateRight?

    -- Right to open a new burn
    public burnEpoch?
    -- Right to burn or replace
    public burnRight*

    -- Ownership right over asse
    private assetOwner* :: Zk64

    genesis       :: spec
                   , data
                   , created
                   , issuedSuppl
                   , reserves {R
                  -> assetOwner*
                   , inflationAl
                   , updateRight
                   , burnEpoch?
                  -- errors whic
                  !! supplyMisma
                   | invalidProo
                   | insufficien
```

```
er+
Owner+

ptional operation, which may not be
lementing the interface

owance+
ract.ProofOfReserves ^ 0..0xFFFF}

llowance*
Owner*

ance
ves

ntract.ProofOfReserves ^ 0..0xFFFF}

age

ntract.ProofOfReserves ^ 0..0xFFFF}

Owner+

age

t?
```

RGB20 AluVM Interface Bytecode

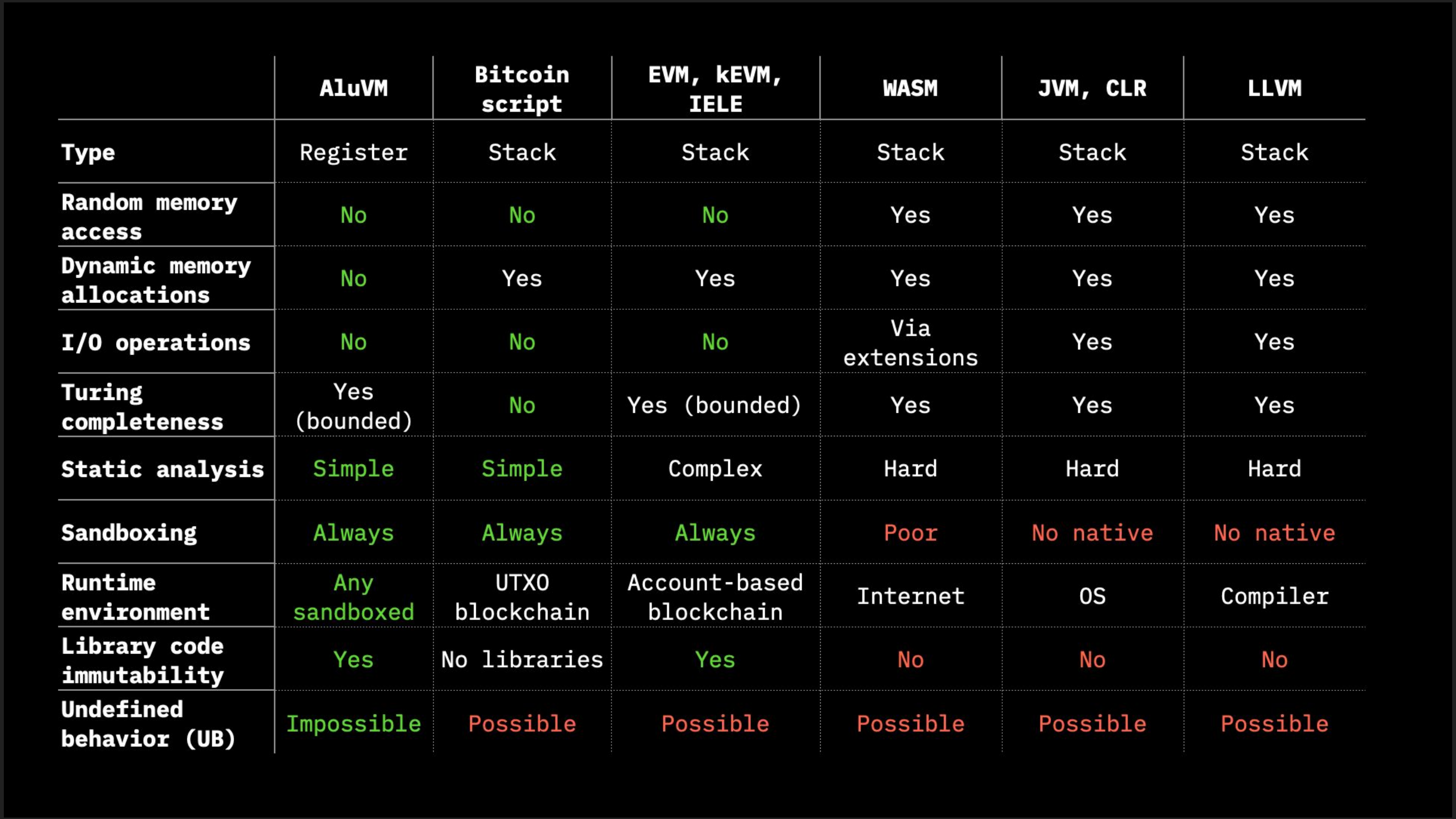


Blame 194 lines (192 loc) · 12 KB

```
-----BEGIN RGB INTERFACE-----
Id: 48hc4i-m9JRcYQA-uUSzwFCK-VNEa9eZf-nhepU8QJ-pqosXS
Mnemonic: laptop-domingo-cool
Name: RGB20

00mM<LNYK03}SV1Ze?Usb#QQOc>#!wSY=~6@jI2b%^Ho0^4h`N6bqMfQQ6em^T$y
fj)VXK2V-(&VRU6=0b;l_K%3oNk|D3>jgMiHNFD$nH%jLO?ZJGP>y9~kh5-NsWMO
n+0UbgMWzazyQWzbJ0`uB~s+m=Ng446R2b*47%pg27;{gB+X>)URWn@!zaBysS0f
>xPWn((=JC(Q18jXtb+QHlu3zu?H+0@$e$59-PgaH8#a%FIAVPj=vQ+04~Y<U5Qj
96u3I`KP|x6K-jit^gQ+!PC!a#7jT+VjUz9FBwl0R(e!Wn%%EVokT?CX$&{lP=%L
(r2YomjWp~)vLf(PH!SxGy3rX00jzRb8}^MPj_x*asmJV0SRJta&AR%Z)0cy0RR9
AVs&zEQfX&sbO8YX0TO9$W^7?}X>V>pY;13LVQyn(0s#O43w3a0VRU6uX=iA30Ra
F200BmOtbsYP^%MO!vmSIsorVgs_HZ-Wn$&XU+C3lhihBeHV{&C-bY)}!0RRC21Y
}`!VE_RD0RRkXb8~fNWK(r;aBO)10RRC21aoj@V*mjF0RRLFVRLh3bWe9~WpV%j0
RR69Vs&zEMR0FpXaE2J0RR$dZf0y@bZKvHL2PVqcVTX0WdHyG|NjehaAaY0Wm0Kp
XmkJo009610|5gB1_VNNa&7?uF^<Zha)%4qQZT7eT575km@BNFKe1k-QjV}dQYWX
O0Ssbwa&Bd0Q+04~Y<U0y009621a)&|WC00cb#iV}X=iA30RRC20S0DubairN0SR
Jta&A&-XJ~W)009610|5gC00l{Nb9H3_0Y-bQfjP1D6a6={9&|;Wh6=Lwa5LJP)N
<z9Js<OmdjSk-b8~fNWK(r;aBO)10RRC20R(k(Wn=*oX>Mk0VRUJ4Zb58pZ+BsCV
`TvV|NjCDVr6b+W@%$-VRCr^3So0|Wpqz>Ze?-+0RR66W_5IRa%BM$X>Mk0VRUJ4
Zb58pZ+BsCV`TsU|Nj640RsdE0SjVfZe?a^V`*V>c?nN&Wo|`qZ)0cy00035b#rB
80SRJta&AR%Z)0cy0096331W3}Zc=GyXmkJp00965Ze@6M0SRJta&AR%Z)0cx009
61009YNb#iV}X=iA322y2iVQpmr009GTWp@Dtb8uy20RRC20R(k(Wn=*hb#P>1bY
)U$XJ~W*0096224;11b#i3^3w3a0VRU6uX=iA30003100037W_5IRa%BfnWpHd^V
`TvWF^<Zha)%4qQZT7eT575km@BNFKe1k-QjV}dQYWXO0S<CyaBN{?Wn@!zaBysS
00962009Jbb7f=!31W3}Zc=GyXmkMp0RRFFVr6b+W@%$-VRCr^3So0|Wpqz>Ze?-
```

AluVM

- Similar to WASM but much more minimal
- VM execution layer compiles to WASM when used in browsers



	AluVM	Bitcoin script	EVM, kEVM, IELE	WASM	JVM, CLR	LLVM
Type	Register	Stack	Stack	Stack	Stack	Stack
Random memory access	No	No	No	Yes	Yes	Yes
Dynamic memory allocations	No	Yes	Yes	Yes	Yes	Yes
I/O operations	No	No	No	Via extensions	Yes	Yes
Turing completeness	Yes (bounded)	No	Yes (bounded)	Yes	Yes	Yes
Static analysis	Simple	Simple	Complex	Hard	Hard	Hard
Sandboxing	Always	Always	Always	Poor	No native	No native
Runtime environment	Any sandboxed	UTXO blockchain	Account-based blockchain	Internet	OS	Compiler
Library code immutability	Yes	No libraries	Yes	No	No	No
Undefined behavior (UB)	Impossible	Possible	Possible	Possible	Possible	Possible

Schema and Interface

- Interfaces define contract semantics
- Schemas define contract logic
- Interface specification will be defined in Contractum (contractum.org)
  - Currently defined in AluVM bytecode
- Schema in AluVM (github.com/AluVM)
- Interface implementation in Rust
- Interface methods called within the wallet

Separation of concerns: the protocols must be designed in a modular and layered way, where each module solves one and only one task. The layers must be well abstracted, meaning that the layers below must be unaware of the structure of the layers above. Such separation of concerns provides a foundation for the protocol interoperability, security, composability and forward-compatibility.

- Section 2.1, Design Goals, RGB Blackpaper - blackpaper.rgb.tech

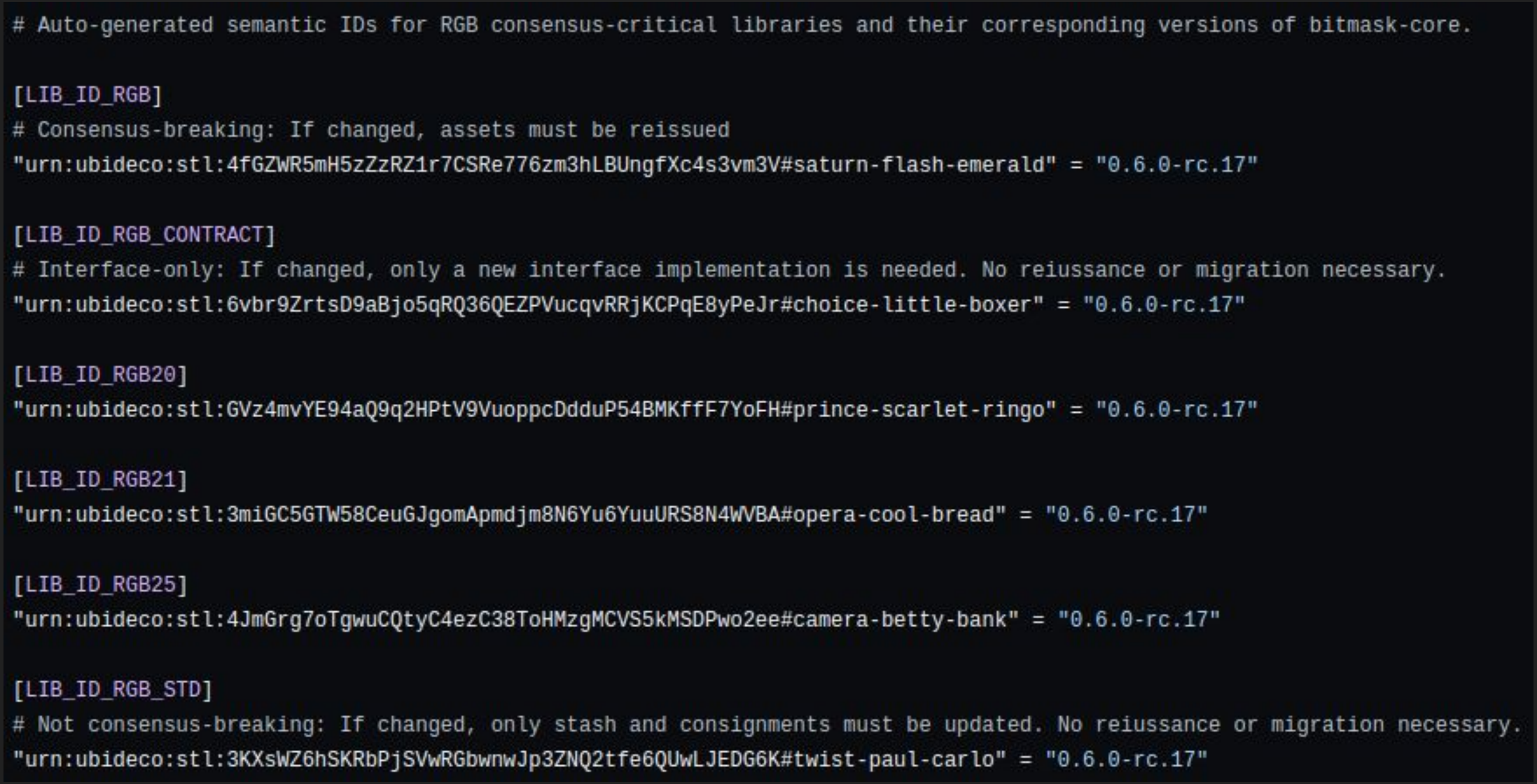
Strict Types

- Used for serialization
- Formal verification and structuring of types
- Rich low-level types (u8, f32, i64, etc, unlike JSON)
- Generates a semantic id to ensure there's no consensus-breaking changes
- Crucial for deterministic client-side validation

Example semantic id for RGB LIB:

```
urn:ubideco:stl:4fGZWR5mH5zZzRZ1r7CSRe776zm3hLBUngfXc4s3vm3V#saturn-flash-emerald
```

https://github.com/diba-io/bitmask-core/blob/development/RGB_LIB_IDs.toml
Generated with this build-script: https://github.com/diba-io/bitmask-core/blob/development/build.rs

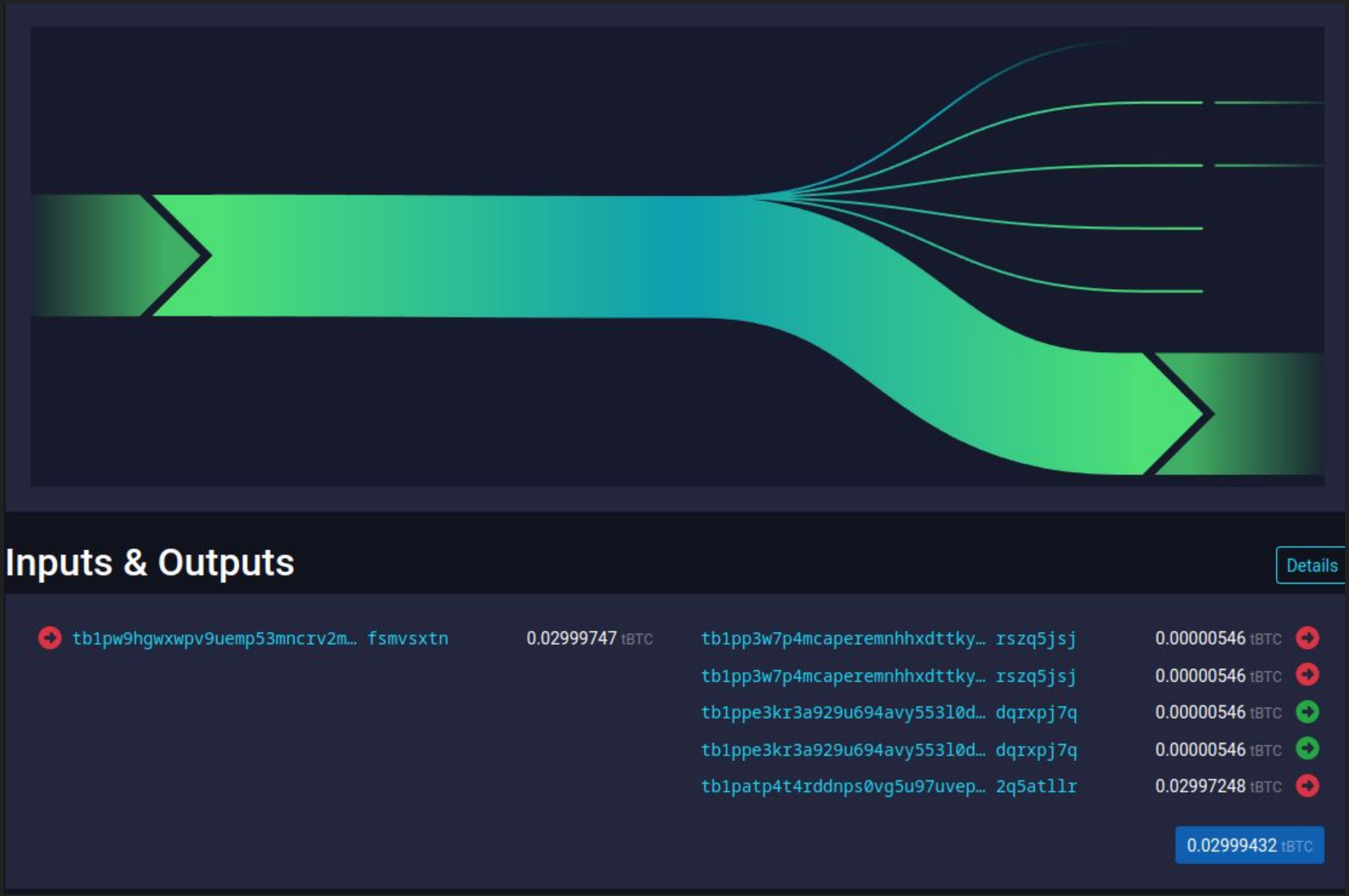


```toml
# Auto-generated semantic IDs for RGB consensus-critical libraries and their corresponding versions of bitmask-core.

[LIB_ID_RGB]
# Consensus-breaking: If changed, assets must be reissued
"urn:ubideco:stl:4fGZWR5mH5zZzRZ1r7CSRe776zm3hLBUngfXc4s3vm3V#saturn-flash-emerald" = "0.6.0-rc.17"

[LIB_ID_RGB_CONTRACT]
# Interface-only: If changed, only a new interface implementation is needed. No reiussance or migration necessary.
"urn:ubideco:stl:6vbr9ZrtsD9aBjo5qRQ36QEZPVucqvRRjKCPqE8yPeJr#choice-little-boxer" = "0.6.0-rc.17"

[LIB_ID_RGB20]
"urn:ubideco:stl:GVz4mvYE94aQ9q2HPtV9VuoppcDdduP54BMKffF7YoFH#prince-scarlet-ringo" = "0.6.0-rc.17"

[LIB_ID_RGB21]
"urn:ubideco:stl:3miGC5GTW58CeuGJgomApmdjm8N6Yu6YuuURS8N4WVBA#opera-cool-bread" = "0.6.0-rc.17"

[LIB_ID_RGB25]
"urn:ubideco:stl:4JmGrg7oTgwuCQtyC4ezC38ToHMzgMCVS5kMSDPwo2ee#camera-betty-bank" = "0.6.0-rc.17"

[LIB_ID_RGB_STD]
# Not consensus-breaking: If changed, only stash and consignments must be updated. No reiussance or migration necessary.
"urn:ubideco:stl:3KXsWZ6hSKRbPjSVwRGbwnwJp3ZNQ2tfe6QUwLJEDG6K#twist-paul-carlo" = "0.6.0-rc.17"
```

Taproot DBCs & MPCs

- Deterministic Bitcoin Commitments
- OP_RETURN commits to a 32-byte hash of contract state
- TapRet commitments embed an OP_RETURN TapScript in a TapLeaf
- MAST hashes TapScripts into a merkle tree
- P2TR public key is an x-only public key derived from merkle root
- Multi-Protocol Commitments allows interoperability with other protocols
    - LNPBP-4 - https://github.com/LNP-BP/LNPBPs/blob/master/lnpbp-0004.md
- Can have over 250,000 different contracts in a single MPC
    - https://github.com/LNP-BP/client_side_validation/pull/132

Funding Transaction creates dust UTXOs

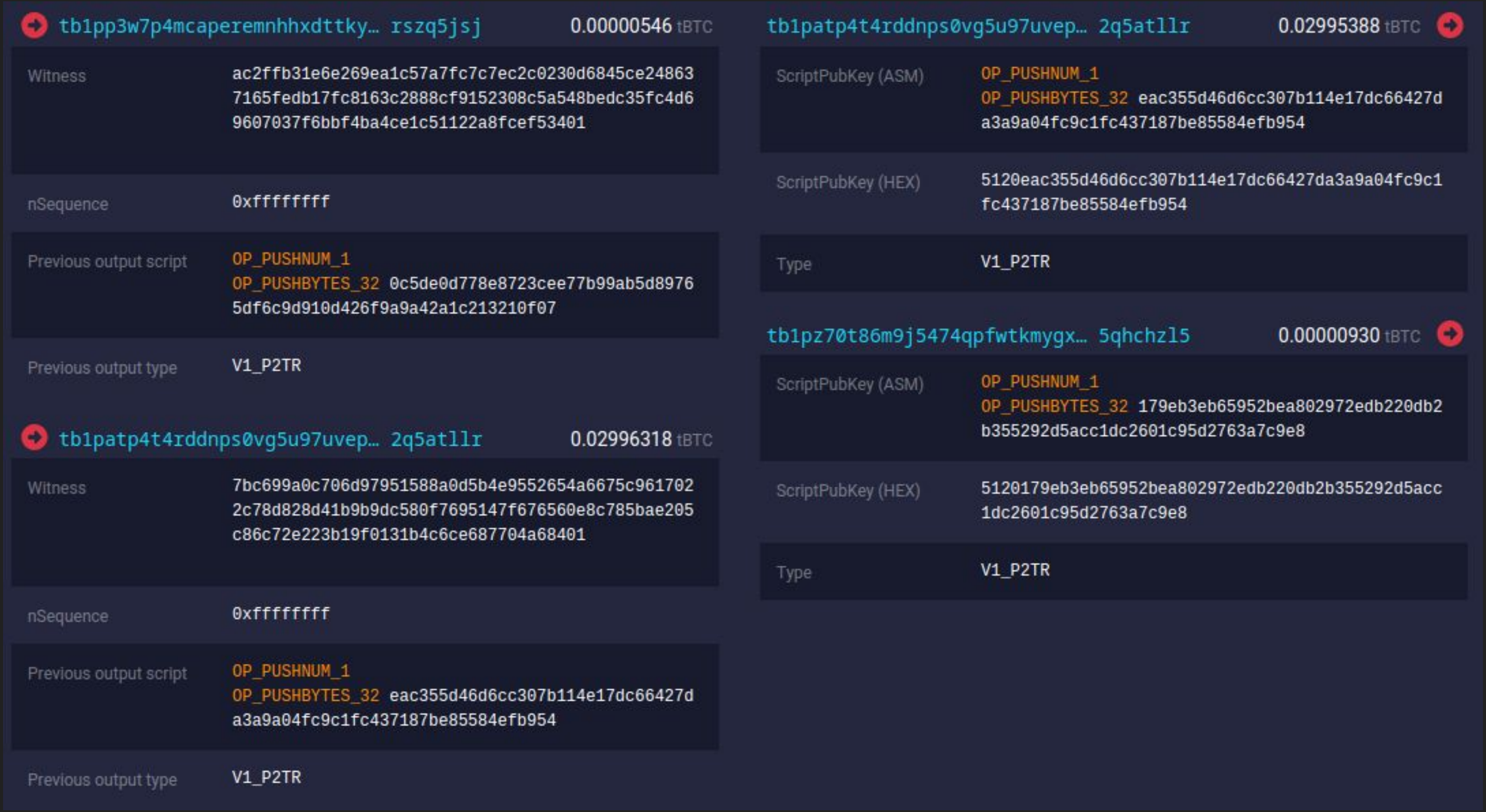


Inputs & Outputs

tb1pw9hgwxwpv9uemp53mncrv2m… fsmvsxtn	0.02999747 tBTC	tb1pp3w7p4mcaperemnhhxdttky… rszq5jsj	0.00000546 tBTC
		tb1pp3w7p4mcaperemnhhxdttky… rszq5jsj	0.00000546 tBTC
		tb1ppe3kr3a929u694avy55310d… dqrxpj7q	0.00000546 tBTC
		tb1ppe3kr3a929u694avy55310d… dqrxpj7q	0.00000546 tBTC
		tb1patp4t4rddnps0vg5u97uvep… 2q5atllr	0.02997248 tBTC

0.02999432 tBTC

Example RGB Transfer Transaction



tb1pp3w7p4mcaperemnhhxdttky… rszq5jsj 0.00000546 tBTC

```
Witness                  ac2ffb31e6e269ea1c57a7fc7c7ec2c0230d6845ce24863
                         7165fedb17fc8163c2888cf9152308c5a548bedc35fc4d6
                         9607037f6bbf4ba4ce1c51122a8fcef53401
nSequence                0xffffffff
Previous output script   OP_PUSHNUM_1
                         OP_PUSHBYTES_32 0c5de0d778e8723cee77b99ab5d8976
                         5df6c9d910d426f9a9a42a1c213210f07
Previous output type     V1_P2TR
```

tb1patp4t4rddnps0vg5u97uvep… 2q5atllr 0.02996318 tBTC

```
Witness                  7bc699a0c706d97951588a0d5b4e9552654a6675c961702
                         2c78d828d41b9b9dc580f7695147f676560e8c785bae205
                         c86c72e223b19f0131b4c6ce687704a68401
nSequence                0xffffffff
Previous output script   OP_PUSHNUM_1
                         OP_PUSHBYTES_32 eac355d46d6cc307b114e17dc66427d
                         a3a9a04fc9c1fc437187be85584efb954
Previous output type     V1_P2TR
```

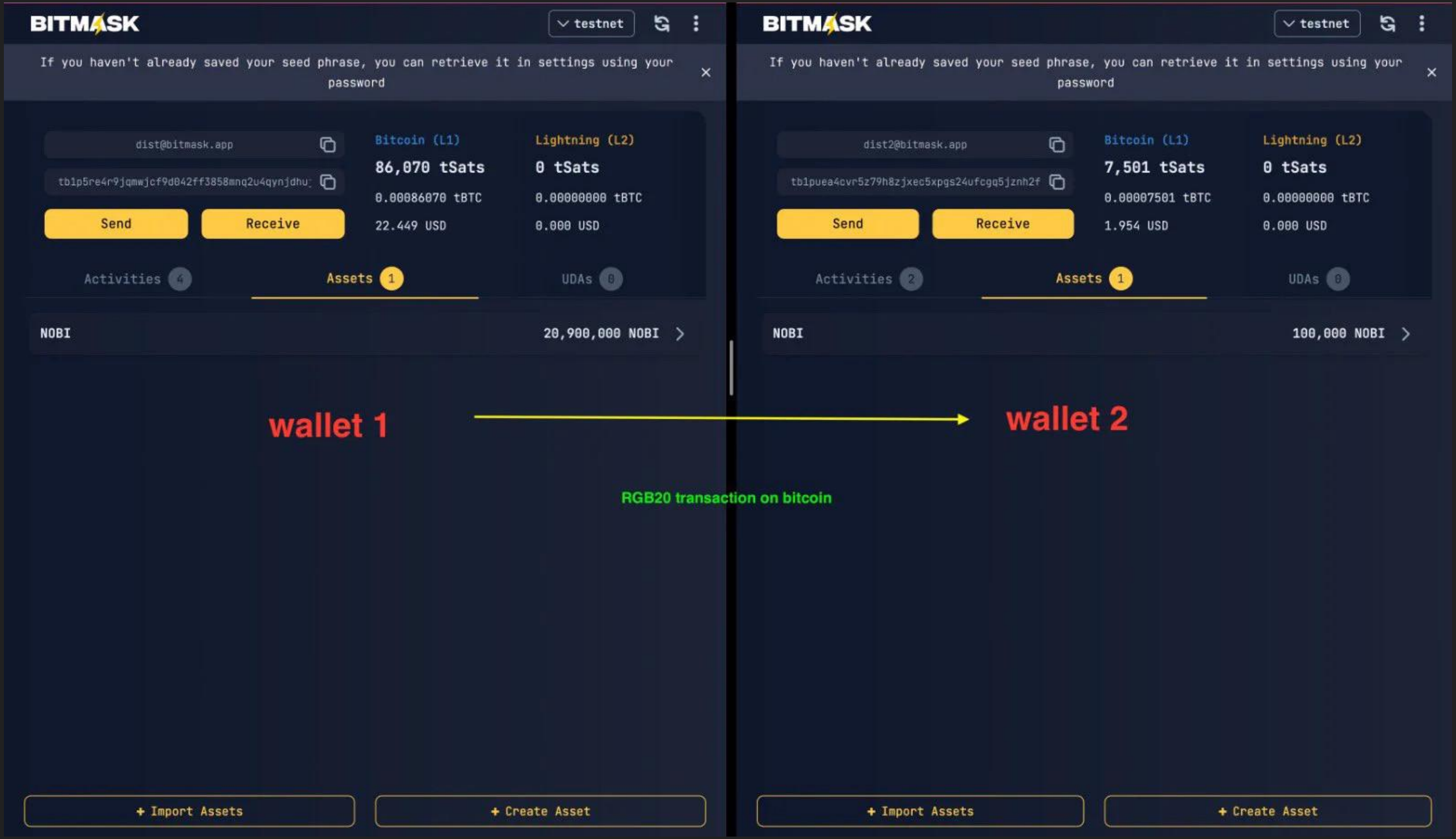
tb1patp4t4rddnps0vg5u97uvep… 2q5atllr 0.02995388 tBTC

```
ScriptPubKey (ASM)   OP_PUSHNUM_1
                     OP_PUSHBYTES_32 eac355d46d6cc307b114e17dc66427d
                     a3a9a04fc9c1fc437187be85584efb954
ScriptPubKey (HEX)   5120eac355d46d6cc307b114e17dc66427da3a9a04fc9c1
                     fc437187be85584efb954
Type                 V1_P2TR
```

tb1pz70t86m9j5474qpfwtkmygx… 5qhchzl5 0.00000930 tBTC

```
ScriptPubKey (ASM)   OP_PUSHNUM_1
                     OP_PUSHBYTES_32 179eb3eb65952bea802972edb220db2
                     b355292d5acc1dc2601c95d2763a7c9e8
ScriptPubKey (HEX)   5120179eb3eb65952bea802972edb220db2b355292d5acc
                     1dc2601c95d2763a7c9e8
Type                 V1_P2TR
```

BitMask Wallet - beta.bitmask.app



BITMASK testnet

If you haven't already saved your seed phrase, you can retrieve it in settings using your password

dist@bitmask.app

tb1p5re4r9jqmwjcf9d042ff3858mnq2u4qynjdhu

Send Receive

Bitcoin (L1)	Lightning (L2)
86,070 tSats	0 tSats
0.00086070 tBTC	0.00000000 tBTC
22.449 USD	0.000 USD

Activities 4 Assets 1 UDAs 0

NOBI 20,900,000 NOBI

+ Import Assets + Create Asset

BITMASK testnet

If you haven't already saved your seed phrase, you can retrieve it in settings using your password

dist2@bitmask.app

tb1puea4cvr5z79h8zjxec5xpgs24ufcgq5jznh2f

Send Receive

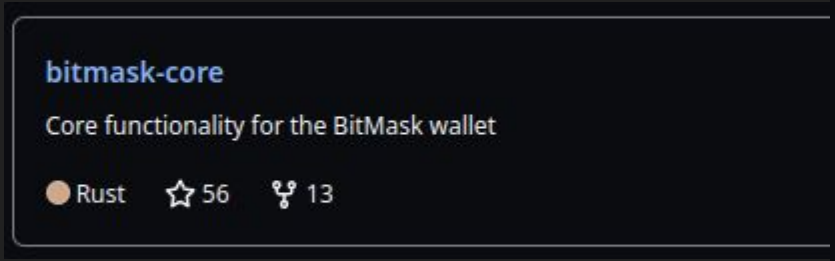
Bitcoin (L1)	Lightning (L2)
7,501 tSats	0 tSats
0.00007501 tBTC	0.00000000 tBTC
1.954 USD	0.000 USD

Activities 2 Assets 1 UDAs 0

NOBI 100,000 NOBI

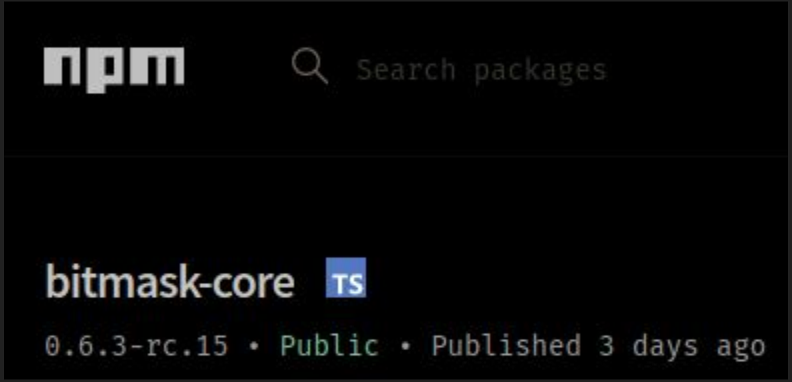
+ Import Assets + Create Asset

Developer Ecosystem


bitmask-core

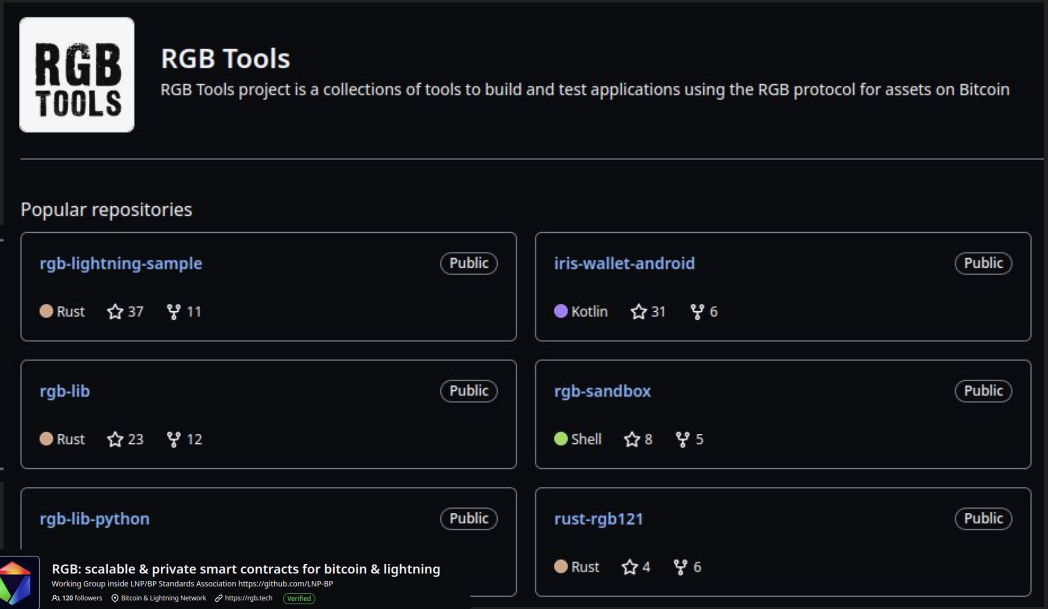
Core functionality for the BitMask wallet

Rust ☆ 56 13



bitmask-core TS

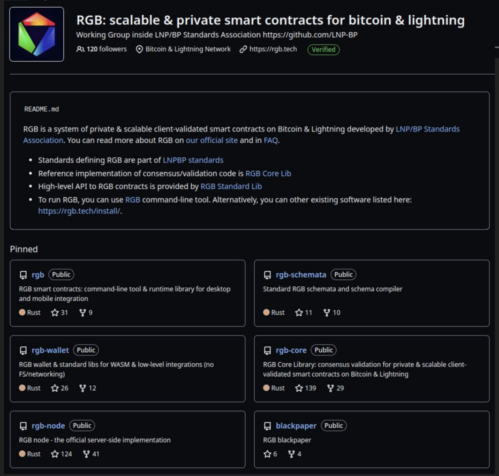
0.6.3-rc.15 • Public • Published 3 days ago


RGB Tools

RGB Tools project is a collections of tools to build and test applications using the RGB protocol for assets on Bitcoin

Popular repositories

rgb-lightning-sample Public Rust ☆ 37 11	iris-wallet-android Public Kotlin ☆ 31 6
rgb-lib Public Rust ☆ 23 12	rgb-sandbox Public Shell ☆ 8 5
rgb-lib-python Public	rust-rgb121 Public Rust ☆ 4 6


RGB: scalable & private smart contracts for bitcoin & lightning

Working Group inside LNP/BP Standards Association https://github.com/LNP-BP

120 followers · Bitcoin & Lightning Network · https://rgb.tech · Verified

README.md

RGB is a system of private & scalable client-validated smart contracts on Bitcoin & Lightning developed by LNP/BP Standards Association. You can read more about RGB on our official site and in FAQ.

- Standards defining RGB are part of LNPBP standards
- Reference implementation of consensus/validation code is RGB Core Lib
- High-level API to RGB contracts is provided by RGB Standard Lib
- To run RGB, you can use RGB command-line tool. Alternatively, you can other existing software listed here: https://rgb.tech/install/.

Pinned

rgb Public RGB smart contracts: command-line tool & runtime library for desktop and mobile integration Rust ☆ 31 9	rgb-schemata Public Standard RGB schemata and schema compiler Rust ☆ 11 10
rgb-wallet Public RGB wallet & standard libs for WASM & low-level integrations (no FS/networking) Rust ☆ 26 12	rgb-core Public RGB Core Library: consensus validation for private & scalable client-validated smart contracts on Bitcoin & Lightning Rust ☆ 139 29
rgb-node Public RGB node - the official server-side implementation Rust ☆ 124 41	blackpaper Public RGB blackpaper ☆ 6 4

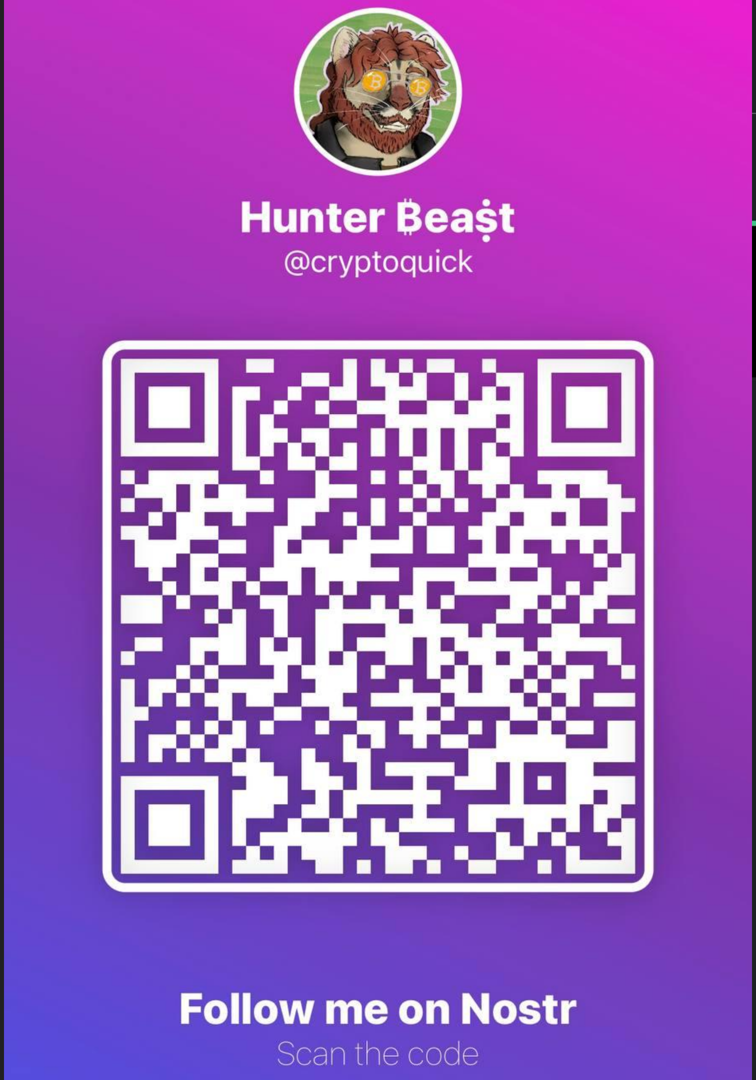
More Resources to Learn About RGB

- rgb.tech - Official RGB Technical site
- blackpaper.rgb.tech - Blackpaper
- rgbfaq.com - FAQ
- standards.lnp-bp.org - Other RGB and LNP/BP standards
- contractum.org - Contractum Spec
- rgb.info - Community site
- rgbex.io - RGB Explorer (more limited than a block explorer)

Thank you!

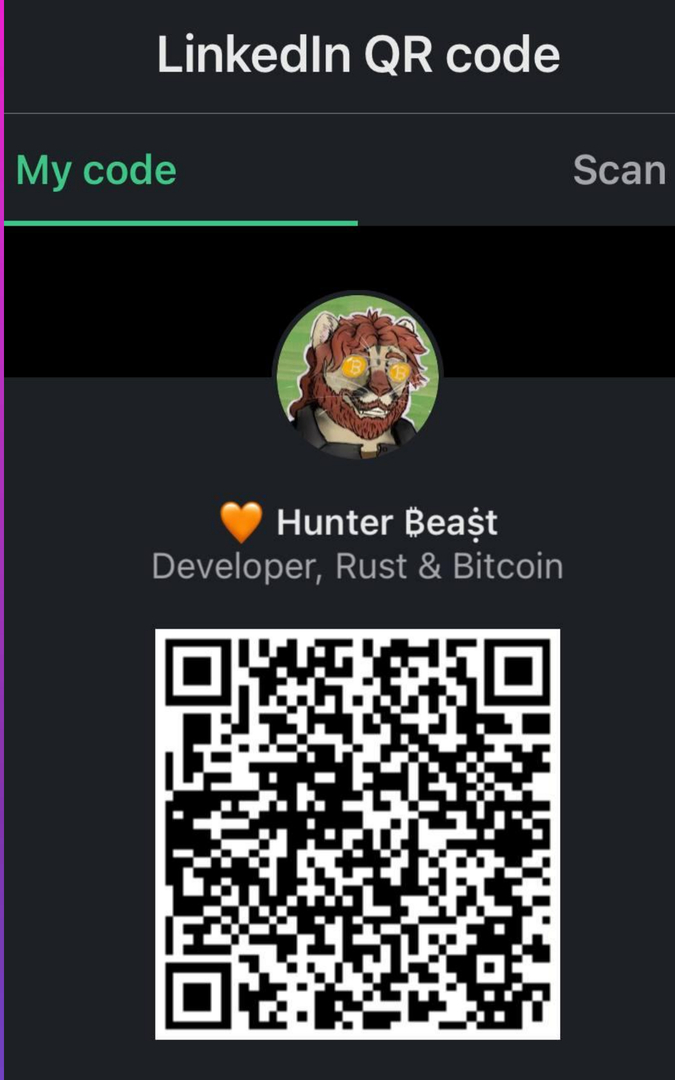
@cryptoquick on:

- X
- GitHub
- Telegram



Hunter Ƀeaṩt
@cryptoquick

Follow me on Nostr
Scan the code



LinkedIn QR code

My code | Scan

🧡 Hunter Ƀeaṩt
Developer, Rust & Bitcoin