

BitMask & RGB

w/ Hunter Beast

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



Hunter Beast

- JavaScript dev turned Rust dev (since 2018)
- Started on shitcoins
 - Ethereum, 2017
 - Filecoin, 2021
- Bitcoin-only since 2022
- RGB -> Bitcoin Maxi
- Joined DIBA in Feb 2022
 - RGB
 - BitMask 0.0.1
- RGB design and review



Now, let's get into it...

Let's start with RGB.

The term RGB—

Red, Green, Blue—

is a reference to...



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?



Why shitcoin?

- People want shitcoins
- We live in a world of shitcoins
 - Food
 - Furniture
 - Houses
- Receipts
- Altruism
- Contracts without authorities
- Might as well do so in the least disruptive way possible



What is RGB?

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



UTXOs

Important to understand:

- Unspent
 - Can only be spent once
- Transaction (TX)
 - Atomic, either happens or it doesn't
- Output
 - Inputs/Outputs, not accounts, like EVM



Single Use Seals

A UTXO proves several things:

- There is a single unique state
- No other UTXO that's the same (esp. after xfer)
 - Non-repudiation
- This state can be spent by its owner
 - "Separation of ownership and state"
- The state is valid
- Double-spend
 - "Single-use seal"
 - Sharding
 - Pedersen commitments
 - Prevent inflation
- Timestamping
- Ordering



Privacy





RGB data is defined by clients
(wallets)

- Contracts
- Beneficiaries
- Participants
- Amounts
- Invoices

None of this goes on-chain.



Example RGB Transfer Transaction

 tb1pp3w7p4mcaperemnhxdttky... rszq5jsj 0.00000546 tBTC	tb1patp4t4rddnps0vg5u97uvep... 2q5atl1r 0.02995388 tBTC 
Witness ac2ffb31e6e269ea1c57a7fc7c7ec2c0230d6845ce24863 7165fedb17fc8163c2888cf9152308c5a548bedc35fc4d6 9607037f6bbf4ba4ce1c51122a8fcef53401	ScriptPubKey (ASM) OP_PUSHDNUM_1 OP_PUSHBYTES_32 eac355d46d6cc307b114e17dc66427d a3a9a04fc9c1fc437187be85584efb954
nSequence 0xffffffff	ScriptPubKey (HEX) 5120eac355d46d6cc307b114e17dc66427da3a9a04fc9c1 fc437187be85584efb954
Previous output script OP_PUSHDNUM_1 OP_PUSHBYTES_32 0c5de0d778e8723cee77b99ab5d8976 5df6c9d910d426f9a9a42a1c213210f07	Type V1_P2TR
Previous output type V1_P2TR	
 tb1patp4t4rddnps0vg5u97uvep... 2q5atl1r 0.02996318 tBTC	tb1pz70t86m9j5474qpfwtkmgyx... 5qhchz15 0.00000930 tBTC 
Witness 7bc699a0c706d97951588a0d5b4e9552654a6675c961702 2c78d828d41b9b9dc580f7695147f676560e8c785bae205 c86c72e223b19f0131b4c6ce687704a68401	ScriptPubKey (ASM) OP_PUSHDNUM_1 OP_PUSHBYTES_32 179eb3eb65952bea802972edb220db2 b355292d5acc1dc2601c95d2763a7c9e8
nSequence 0xffffffff	ScriptPubKey (HEX) 5120179eb3eb65952bea802972edb220db2b355292d5acc 1dc2601c95d2763a7c9e8
Previous output script OP_PUSHDNUM_1 OP_PUSHBYTES_32 eac355d46d6cc307b114e17dc66427d a3a9a04fc9c1fc437187be85584efb954	Type V1_P2TR
Previous output type V1_P2TR	

RGB-20

- Token metadata is specified
 - (name, supply, precision) <- "global state"
 - Issued against a UTXO <- "owned state"
- Consignment
 - A change in ownership of a contract
- Transfers
 - Blinded UTXO
 - Invoice
 - Transfer Consignment
 - Relay Consignment
 - Accept Consignment
 - Stash
 - Stock

```
interface: RGB20

globals:
  spec:
    naming:
      ticker: DIBA
      name: Amazing Diba Contract
      details: ~
    precision: 2
  data:
    terms: >
      DONT TRUST, VERIFY.
    media: ~
    issuedSupply: 10000000000000
    created: 1687969158

assignments:
  assetOwner:
    seal: tapret1st:c9a86c99127f1b2d1ff495c238f13069ac881ec9527905016122d11d85b19b61:1
    amount: 10000000000000
```

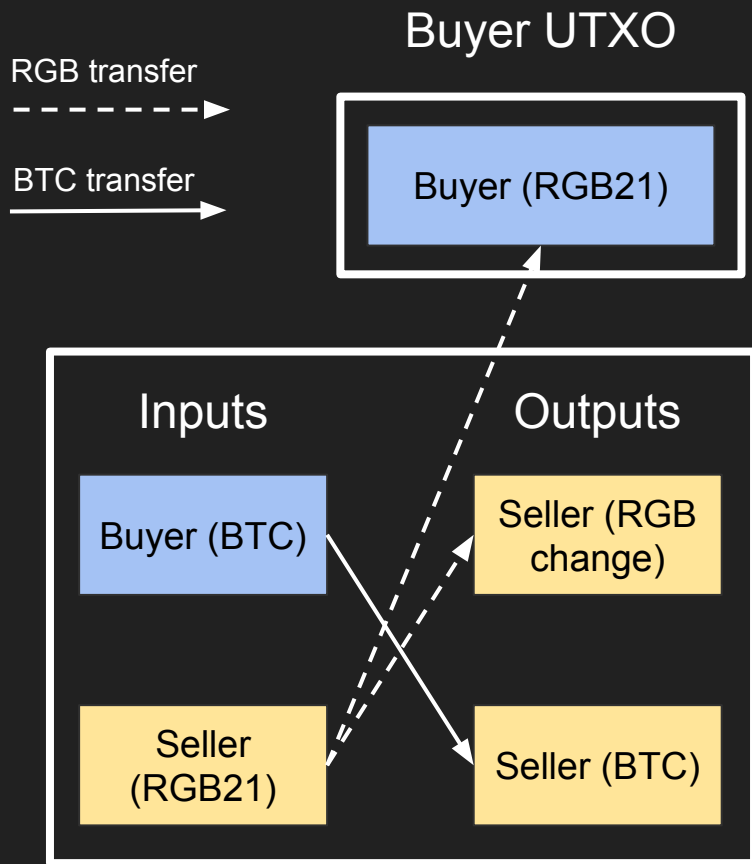


RGB-21

- Similar to RGB-20
 - Both use LNPBP-31 std lib
- Unique Digital Asset
- Equivalent:
 - Supply of 1
 - Precision of 0
- Attachments
 - MIME type (e.g., image/jpeg)
 - URI
- Consignment (16MB)



RGB Atomic Swap TX



RGB20 Contractum Interface Specification

```
-- Defined by LNPBP-31 standard in 'RGBContract.sty' file
import urn:ubideco:stl:6vbr9ZrtsD9aBj05qRQ36QEzPVucqvRRjKCPqE8yPeJr#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
```


RGB

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

```
interface RGB20
  -- Asset specification containing
  global spec :: RGBContract.D
```

```
  -- Contract data and creation
  -- not be changeable by the
  global data :: RGBContract.C
  global created :: RGBContract
```

```
  -- State which accumulates a
  global issuedSupply+ :: RGBContract
  -- State which accumulates a
  global burnedSupply* :: RGBContract
  -- State which accumulates a
  global replacedSupply* :: RGBContract
```

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

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

```
-- 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
```

ification

```
er+
Owner+
```

optional operation, which may not be
implementing the interface

```
allowance+
ract.ProofOfReserves ^ 0..0xFFFF}
allowance*
Owner*
```

```
ance
ves
```

```
contract.ProofOfReserves ^ 0..0xFFFF}
```

```
age
```

```
contract.ProofOfReserves ^ 0..0xFFFF}
Owner+
```

```
age
```

```
t?
```

RGB20 Strict Encoded Base85 Interface

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#QQ0c>#!wSY=-6@jI2b%^Ho0^4h`N6bqMfQQ6em^TSy
fj)VXK2V-(&VRU6=0b;l_K%3oNk|D3>jgMiHNFD$NH%jL0?ZJGP>y9-kh5-NswMO
n+0UbgMwzazyQWzbJ0`uB~s+m=Ng446R2b*47%pg27;{gB+X>)URWn@!zaBysS0f
>xPWN( (=JC(Q18jXtb+QHlu3zu?H+00$e$59-PgaH8#a%FIAVPj=vQ+04~Y<U5Qj
96u3I`KP|x6K-jit^gQ+!PC!a#7jT+VjUz9FBw10R(e!Wn%%EVokT?CX$&{1P=%L
(r2YomjWp~)vLf(PH!SxGy3rX00jzRb8}^MPj_x*asmJV0SRJta&AR%Z)0cy0RR9
AVs&zEQFX&sb08YX0T09$W^7?)X>V>pY;13LVQyn(0s#043w3a0VRU6uX=iA30Ra
F200Bm0tbsYP^%M0!vmSIsoVgs_HZ-Wn$&XU+C3lhihBeHV{&C-bY)}!0RRc21Y
j`!VE_RD0RRkXb8~fNWK(r;aB0)10RRc21aoj@V*mjF0RRlFVRLh3bWe9~WpV%j0
RR69Vs&zEMR0FpXaE2J0RR$dZf0y@bZKvHL2PVqcVTX0WdHyG|NjehaAaY0Wm0Kp
XmkJo009610|5gB1_VNna&7?uF^<Zha)%4qQZT7eT575km@BNFKe1k-QjV}dQYWX
00Ssbwa&Bd0Q+04~Y<U0y009621a)&|WC00cb#iV}X=iA30RRc20S0DubairN0SR
Jta&A&-XJ-W)009610|5gC001{Nb9H3_0Y-bQfjP1D6a6={9&|;Wh6=Lwa5LJP)N
<z9Js<OmdjSk-b8~fNWK(r;aB0)10RRc20R(k(Wn=*oX>Mk0VRUJ4Zb58pZ+BscV
`TvV|NjCDVr6b+W00$-VRCr^3So0|Wpqz>Ze?-+0RR66W_5IRa%BM$X>Mk0VRUJ4
Zb58pZ+BscV`TsU|Nj640RsdE0SjVfZe?a^V`*V>c?nN&Wo|`qZ)0cy00035b#rB
80SRJta&AR%Z)0cy0096331W3}Zc=GyXmkJp00965Ze06M0SRJta&AR%Z)0cx009
61009YNb#iV}X=iA322y2iVQpmr009GTWp@Dtb8uy20RRc20R(k(Wn=*hb#P>1bY
)USXJ-W*0096224;11b#13^3w3a0VRU6uX=iA30003100037W_5IRa%BfnWpHd^V
`TvWf^<Zha)%4qQZT7eT575km@BNFKe1k-QjV}dQYWX00S<CyaBN{?Wn@!zaBysS
009620001hh7f=!31W317c=GvXmkMn0RRcFVr6b+W00$-VRCr^3So0|Wpqz>Ze?-
```

Schema and Interface

- Interfaces define contract semantics
- Schemas define contract logic
- Compiled to Strict Encoded ASCII-Armored Base85
- Interface specification will be defined in Contractum (contractum.org)
- 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)
- Types like u4, u5, u128, u256, u512, u1024 unlike Rust
- Type Confinement
- 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:4fGZWR5mH5zZzRZ1r7CSRe776zm3hLBUngefXc4s3vm3V#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>

```
# 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:4fGZWR5mH5zZzRZ1r7CSRe776zm3hLBUn9fXc4s3vm3V#saturn-flash-emerald" = "0.6.0-rc.17"
```

```
[LIB_ID_RGB_CONTRACT]
```

```
# Interface-only: If changed, only a new interface implementation is needed. No reissuance or migration necessary.
```

```
"urn:ubideco:stl:6vbr9ZrtsD9aBjo5qRQ36QEZPVucqvRRjKCPqE8yPeJr#choice-little-boxer" = "0.6.0-rc.17"
```

```
[LIB_ID_RGB20]
```

```
"urn:ubideco:stl:GVz4mvYE94aQ9q2HPTv9VuoppCdduP54BMKffF7YoFH#prince-scarlet-ringo" = "0.6.0-rc.17"
```

```
[LIB_ID_RGB21]
```

```
"urn:ubideco:stl:3miGC5GTW58CeugGJgomApmdjm8N6Yu6YuuURS8N4WVBA#opera-cool-bread" = "0.6.0-rc.17"
```

```
[LIB_ID_RGB25]
```

```
"urn:ubideco:stl:4JmGrg7oTgwuQQtyC4ezC38ToHMzgMCVS5kMSDPwo2ee#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 reissuance or migration necessary.
```

```
"urn:ubideco:stl:3KXsWZ6hSKRbPjSVwRGbwnwJp3ZNQ2tfe6QUwLJEDG6K#twist-paul-carlo" = "0.6.0-rc.17"
```

Taproot DBCs

- Deterministic Bitcoin Commitments
- OP_RETURN commits to a 32-byte hash of contract state transition bundle
- 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 tweaked by merkle root

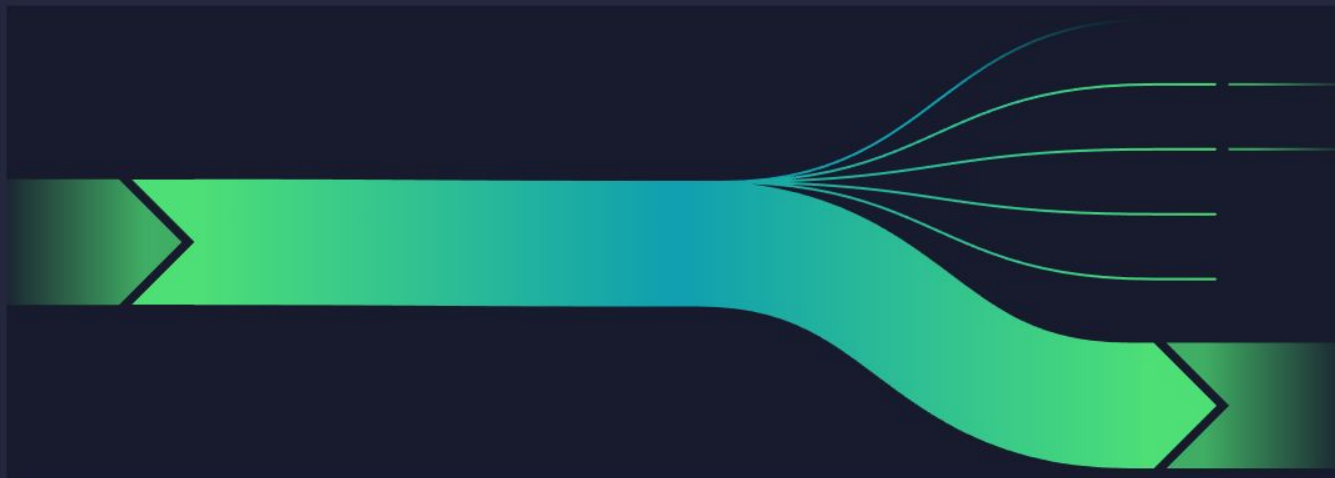


Taproot MPCs

- Multi-Protocol Commitments allows interoperability with other protocols
 - [LNPBP-4](#)
- Uses TapTrees to commit to multiple DBCs
- [Can have over 250,000 different contracts in a single MPC](#)









Funding Transaction creates dust UTXOs



Inputs & Outputs

[Details](#)

 tb1pw9hgwxwpv9uemp53mncrv2m... fsmvsxtn	0.02999747 tBTC	tb1pp3w7p4mcaperemnhhxdttky... rszq5jsj	0.00000546 tBTC	
		tb1pp3w7p4mcaperemnhhxdttky... rszq5jsj	0.00000546 tBTC	
		tb1ppe3kr3a929u694avy553l0d... dqrxpj7q	0.00000546 tBTC	
		tb1ppe3kr3a929u694avy553l0d... dqrxpj7q	0.00000546 tBTC	
		tb1patp4t4rddnps0vg5u97uvep... 2q5atl1r	0.02997248 tBTC	

0.02999432 tBTC

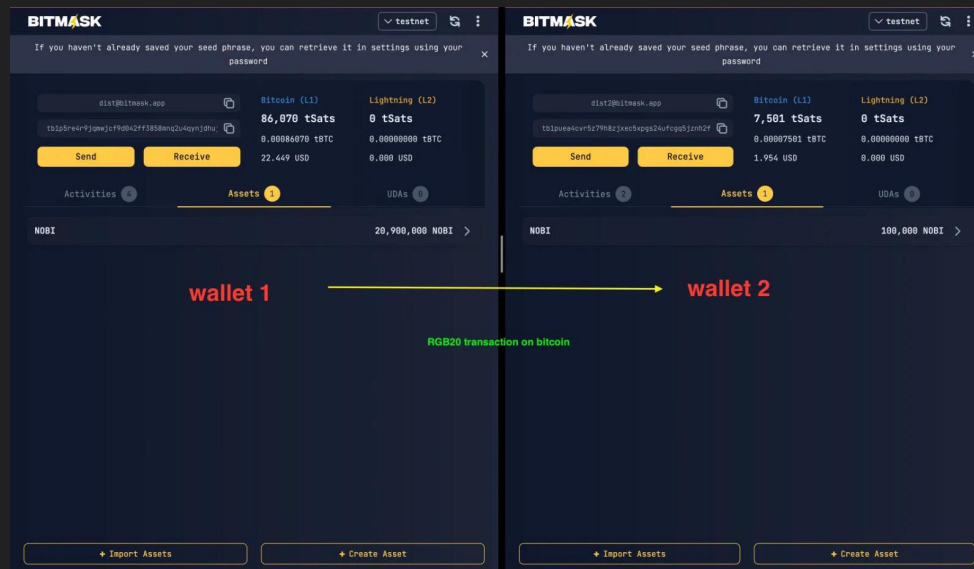
DIBA

- Digital Bitcoin Assets
- Marketplace for digital assets with a focus on RGB assets
- RGB21 - Unique Digital Assets (not NFTs)
- "Phygital" assets
- We also make a wallet...



BitMask

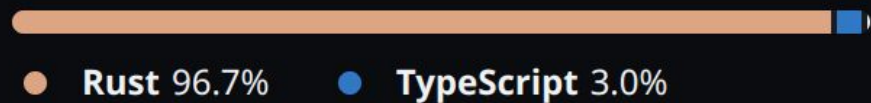
- Noncustodial Bitcoin Taproot (bc1p)
- RGB utility wallet
 - Layer 1 only
 - RGB21 Transfers
 - RGB21 Swaps
 - RGB20 Swaps
 - Prime
- Custodial Lightning wallet
 - Supports usernames
 - Lightning Address (example: hunterbeast@bitmask.app)
 - NIP-05 ID
 - Uses Kollider LNDHubX (Rust implementation of LNDHub)
- Nostr keys
 - Contacts soon



bitmask-core

- BDK
- PDK
- NDK
- (No LDK)
- LNDHubX
- Deezy
- Carbonado
- RGB

Languages



λ scc bitmask-core

Language	Files	Lines	Blanks	Comments	Code	Complexity
Rust	75	24414	3069	1049	20296	847
Shell	8	70	4	8	58	3
TypeScript	7	1358	183	210	965	17
JSON	5	165	1	0	164	0
TOML	4	181	19	10	152	1
Dockerfile	2	43	12	3	28	3
License	2	39	6	0	33	0
YAML	2	228	42	4	182	0
gitignore	2	14	0	0	14	0
Docker ignore	1	6	0	0	6	0
Markdown	1	91	33	0	58	0
Total	109	26609	3369	1284	21956	871

Estimated Cost to Develop (organic) \$692,191

Estimated Schedule Effort (organic) 11.96 months

Estimated People Required (organic) 5.14

Processed 1085776 bytes, 1.086 megabytes (SI)

Keys

- Encrypted using Argon2id hash and AES-GCM 256
- Derived using secp256k1, BDK, and NDK crates

```
export interface Vault {  
  mnemonic: string;  
  private: PrivateWalletData;  
  public: PublicWalletData;  
}
```

bitmask-core / lib / web / bitcoin.ts

Code

Blame

213 lines (187 loc) · 4.79 KB

```
98  
99 // Core type interfaces based on structs defined within the bitmask-core Rust crate:  
100 // https://github.com/diba-io/bitmask-core/blob/development/src/structs.rs  
101  
102 export interface PrivateWalletData {  
103   xprvk: string;  
104   btcDescriptorXprv: string;  
105   btcChangeDescriptorXprv: string;  
106   rgbAssetsDescriptorXprv: string;  
107   rgbUdasDescriptorXprv: string;  
108   nostrPrv: string;  
109   nostrNsec: string;  
110 }  
  
export interface PublicWalletData {  
  xpub: string;  
  xpubkh: string;  
  watcherXpub: string;  
  btcDescriptorXpub: string;  
  btcChangeDescriptorXpub: string;  
  rgbAssetsDescriptorXpub: string;  
  rgbUdasDescriptorXpub: string;  
  nostrPub: string;  
  nostrNpub: string;  
}
```

```
import * as BMC from "../bitmask_core";
```

```
export const hashPassword = (password: string) => BMC.hash_password(password);
```

```
export const decryptWallet = async (
```

```
  hash: string,  
  encryptedDescriptors: string
```

```
): Promise<Vault> =>
```

```
  JSON.parse(await BMC.decrypt_wallet(hash, encryptedDescriptors));
```


Carbonado (1/3) - What is Carbonado?

Crate **carbonado** 

[source](#) · [-]

Carbonado: An apocalypse-resistant data storage format for the truly paranoid.

Carbonado is an archival format for encrypted, durable, compressed, provably replicated consensus-critical data, without need for a blockchain or powerful hardware. Decoding and encoding can be done in the browser through WebAssembly, built into remote nodes on P2P networks, kept on S3-compatible cloud storage, or locally on-disk as a single highly portable flat file container format.

Modules

constants	For details on Carbonado formats and their uses, see the Carbonado Format bitmask constant .
error	Error types
file	File helper methods.
structs	See structs::EncodeInfo for various statistics gathered in the encoding step.
utils	Various utilities to assist with Carbonado encoding steps.

Functions

decode	Decode data from Carbonado format in reverse order: <code>bao</code> -> <code>zfec</code> -> <code>ecies</code> -> <code>snap</code>
encode	Encode data into Carbonado format, performing compression, encryption, adding error correction codes verification encoding, in that order.
extract_slice	Extract a 1KB slice of a Bao stream at a specific index.
scrub	Scrub <code>zfec</code> -encoded data, correcting flipped bits using forward error correction codes. Returns an error w valid data cannot be provided, or data is already valid.
verify_slice	Verify a number of 1KB slices of a Bao stream starting at a specific index.



Carbonado (2/3) - What does Carbonado do?

Encoding:

1. Snappy compression
2. secp256k1 AES-GCM 256 encryption
3. Zfec Forward Error Correction
4. Bao Stream Verification
 - a. Blake3

Decoding:

1. Verify stream
2. Error correction
 - a. If verify succeeds, strip
 - b. If verify fails, scrub
3. Decryption
4. Decompression

Stream Verification



1TB

Carbonado (3/3) - How do we use Carbonado?

- Two methods are exposed by bitmask-core: store, retrieve
- Nostr Hex Secret Key is used for both
- Carbonado encodes Stock
 - in-memory
 - serverless
 - end-to-end encrypted
 - stored on multiple servers
 - automerge CRDTs

```
import * as BMC from "../bitmask_core";

export const store = async (
  nostrHexSk: string,
  data: Uint8Array,
  force: boolean,
  name?: string,
  meta?: Uint8Array
): Promise<void> => BMC.store(nostrHexSk, name || "", data, force, meta);

export const retrieve = (
  nostrHexSk: string,
  lookup: string
): Promise<Uint8Array> => BMC.retrieve(nostrHexSk, lookup);
```


BitMask Roadmap

- LN Swaps
- RGB Swaps
- Transfer Batching
- Iris Compatibility
- Nostr Contacts
- Nostr Wallet Connect
- BIP-21 support
- WebBTC & WebLN (& WebRGB?)
- Teleburning Inscriptions
- Marketplace Launch on 11/11
- Prime, Radiant, Abraxas



Prime

- Alternate L1
 - Anchored to Bitcoin
 - Mining Seal
 - ANYONECANSPEND
 - Tapret
- Proof Merkle Tree
- Seals only (RGBTC)
- Unbounded TPS
- Node interactivity
 - 6GB/UTXO/year
 - Prove to spend
- Requires no changes to Bitcoin



Radiant

- Prometheus state channels
- Multisig between randomly chosen parties
- Arbitrates disputes
- Staking
- Peg-in issues RGB30 RGBTC
 - RGB30 - Decentralized Issuance
- Redeem tokens to peg-out
 - New UTXOs are made
- Requires no changes to Bitcoin



Abraxas


- Alternate L1
- Non-interactive
 - Data availability
- 1 million seals per block
 - Anchored to Bitcoin
 - Every 10 minutes
- ZK-STARKs
- 20-40GB per year
- Spam-resistant
 - Seals-only
- Fees paid in on-chain ABTC
- Requires no changes to Bitcoin




Developer Ecosystem

bitmask-core

Core functionality for the BitMask wallet

 Rust  56  13



 Search packages

bitmask-core

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 LN/PS Standards Association <https://github.com/LN-PS>

 120 followers  Bitcoin & Lightning Network  <https://rgb.tech> 

README.md

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

- Standards defining RGB are part of LN/PS 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  27  3

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 programming)

 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

Resources to Learn More 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 traditional block explorer)



Thank you!

@cryptoquick on:

- X
- GitHub
- Telegram



Hunter Beşt
@cryptoquick



Follow me on Nostr

Scan the code

LinkedIn QR code

My code

Scan



♥ Hunter Beşt
Developer, Rust & Bitcoin

