RGB smart contract advancements

Fungible assets & NFTs, including digital art, identity, naming, reputation

Implementation progress & updates

Core lib progress

- Improving and extending general API to work with state transitions etc (`StateUnit` type)
- Disclosures completed
- Data containers API (will be discussed later this call)
- Virtualization advancements (will be discussed later this call)

RGB20 progress (fungible assets)

- Secondary issuance
- Burn & replacement epochs and procedures

RGB asset information is a special consignment containing only non-transfer types of state transitions (issue, burn, replacement, renomination)

- Subschema for simplified RGB20 asset without replacement procedure

NFTs for digital art & virtual items

RGB-22 schema

RGB21: NFTs for digital art & virtual items

- Simplification of the schemata & transfer rules
- Ability to add medium resources with data containers

RGB NFTs advancements over other NFT tech

- Data storage:
  - no need in costly blockchain storage for NFT
  - no need in IPFS (which is also unencrypted)
  - no need in special data management solution
  - no need in trusting websites to keep the data
- Embedded DRM-type encryption and ownership management
- “Engraving” procedure
- Infrastructure for backups utilizing Lightning Network
- More ways to monetize content
  (sell not the NFT itself, but access to the content, multiple times)

Making DRM anarchic & trustless

- Media resources are kept within encrypted containers
- Differentiation of ownership rights and usage rights
  - Usage rights are controlled by access to the encryption key
  - Ownership is defined by RGB-based single-use-seal (part of RGB22)
- Copyright (“right to copy”) is void in tech world;
  there is “right to use” (useright), author right and ownership right

Questions to discuss

- Use of MIME (Media) types for content type description
- Encryption procedure
  - Simple AES-based encryption (generic for all of the world)
  - Asymmetric encryption, specific for item user (“tagged DRM”)
- Other possible requirements for NFT: make sure we are not missing something

RGB: Introducing data containers

keeping large-size blobs of _non-validated_ data

Use cases

- NFT media resources
- RGB20 large-size Ricardian contracts
- RGB20 asset meta-information
- Details for proofs about burn-and-replace procedures
- Storm-stored data – and Bifrost consignments storage
- Resources for computing

Rules

- RGB Schema/simplicity does not have access to the contained data
- Bitcoin witness-type of extension

Bifrost protocol

operating as a part of lightning network

Bifrost

- RGB-related LN P2P message extensions for data transfer over LN
- Depends on data container extensions for RGB contracts
- Not the same as RGB-over-LN
  (it requires other non-Bifrost extensions and is schema-specific)
- Not the same as Storm
  (it is not RGB-related and can’t understand the structure of RGB data)
- Will be important part of
  - RGB20 payments when receiver is offline
  - NFT media distribution
  - Identity & decentralized naming system
  - Required for building DEX (asset info)

Tasks solved by Bifrost

- Passing RGB consignments for transfers
- Spreading RGB genesis, secondary issuance & supply asset data (genesis & consignments)
- Asset meta-information (issuer signature, identity, media resources) – in form of data containers
- Storage of data containers, including encrypted consignments for third parties

Early draft: https://github.com/LNP-BP/LNPBPs/pull/97/files

Virtualization of RGB validation rules

RGB client-side-validation

Applies to *consignments* and (in restricted form) to *disclosures*

- **Conservative (rule-based)**: VM is not required, since it is
  “once written, never changed” and applies to all future schemata and forms of RGB contracts
  - Validation of internal node graph consistency
    (presence of all referenced contract graph nodes)
  - Validation against schema structure
  - Validation against single-use-seal commitment medium – prevention of double-spending
    (blockchain, state channel transaction graph structure)
- **Variative (script-based)**: planned to be implemented with Simplicity VM,
  but currently written as embedded rust procedures (not in form of VM)
  - RGB20: validation of inflation, issuance volumes
  - RGB21,22 (NFT, identity): validation of 1-to-1 NFT transfer rules for operations combining multiple NFTs/identities

Virtualization of script-based validations

- Virtualization helps:
  - distinguishing schema-specific code from other code
  - isolating state of the contract from other logic of RGB Core library
  - may allow creation of custom schemata by undefended devs
- Simplicity feels months, if not year(s) away from being ready for production

Virtualization selection criteria

	Status quo	Rust-based custom VM	WASM	Simplicity
Code audibility	Poor	Average	Good	Excellent (formally provable)
Safety	Endianess is tricky		Good	Excellent
Cryptographic primitives	Grin-based, hard migration	Grin-based, smooth migration	Not known, but smooth migration	Not ready
VM Implementation effort	None	Very high	High	Extreme high
Schema validation implementation effort	Medium	Low	High	Extreme high
Can be used by other schema devs	No	Hardly	Easily	Not really

Scoring

	Status quo	Rust-based custom VM	WASM	Simplicity
Security	0	3	5	8
Implementation complexity	3	1	2	0
Extensibility	7	5	4	0
Overall	**10**	9	**11**	**8**