Unstoppable Money

Tries to conserve network, disk space and RAM and only use it for mission critical stuff.

Adding anything non-financial can cause friction

Showing earlier attempts, then suggested approach (client side validation + pay to contract)

In particular: list of coins (UTXO set) is ideally kept in RAM, to quickly check new blocks, earlier schemes didn't account for this

http://www.righto.com/2014/02/ascii-bernanke-wikileaks-photographs.html

Nelson Mandela (1918-2013)
"I am fundamentally an optimist. Whether that comes from nature or nurture, I cannot say. Part of being optimistic is keeping one's head pointed toward the sun, one's feet moving forward. There were many dark moments when my faith in humanity was sorely tested, but I would not and could not give myself up to despair. That way lays defeat and death."
"I learned that courage was not the absence of fear, but the triumph over it. The brave man is not he who does not feel afraid, but he who conquers that fear."
"Difficulties break some men but make others. No axe is sharp enough to cut the soul of a sinner who keeps on trying,

The data is stored in the blockchain by encoding hex values into the addresses. Below is an excerpt of one of the transactions storing the Mandela information. In this transaction, tiny amounts of bitcoins are being sent to fake addresses such as 15gHNr4TCKmhHDEG31L2XFNvpnEcnPSQvd. This address is stored in the blockchain as hex 334E656C736F6E2D4D616E64656C612E6A70673F. If you convert those hex bytes to Unicode, you get the string 3Nelson-Mandela.jpg?, representing the image filename. Similarly, the following addresses encode the data for the image. Thus, text, images, and other content can be stored in Bitcoin by using the right fake addresses.

€0.20 per 20 bytes -> €10 / kb, but that's paying 26 sat / byte. With low fees: ~€0.40 per kb.

Bitcoin doesn't know Mandela.
Addresses look real, someone could spend them any moment.
Probably not, but can't take the risk.

Spammers gonna spam

Relay OP_RETURN data TxOut as standard transaction type. #2738

Merged **gavinandresen** merged 1 commit into `bitcoin:master` from `jgarzik:op_return` on 22 Oct 2013

Conversation 45 | Commits 1 | Checks 0 | Files changed 5

jgarzik commented on 4 Jun 2013 — Contributor

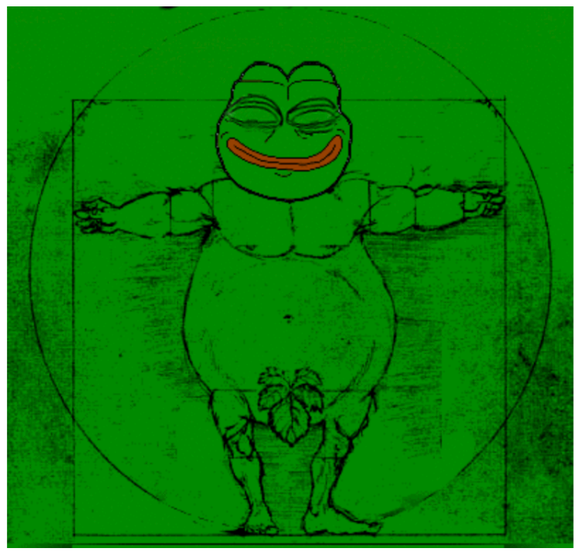
Add new standard transaction type, that permits small amount of data to be attached to a transaction, in the form of an additional TxOut that is provably prunable.

introduced in 2013

permit: this method was not practical before, unless you knew a miner

provably prunable: download, process, discard

I'll save you the analogy Luke-Jr used



RPEPEGOLDENR Asset Information

Asset Name	Description
RPEPEGOLDENR	RPEPEGOLDENR

DEX Trade Price	Market Cap	Total Supply	Divisible	Locked
000000 XCP ($4.04)	$1,270.09	314	False	True

Issuer	
SGbbcWHV8Xy17E9ZyEGW2nKsdRWwmNU	15DSGbbcWHV8Xy17E9ZyEGW2nKsdRWwmNU

Asset Reputation

Current Rating	Last 30 Days	6 months ago	1 Year ago
NA	NA	NA	NA

Additional Information | Dividends | Holders | Issuances | Markets | Orders | Sends | Subassets | Unconfirmed

per page — Page 1 of 1 — 2 res

Block	Time	Asset	Quantity	Locked	Transfer	Fee
434,311	2 years ago	RPEPEGOLDENR	0	True	False	0.00000000 XCP
434,225	2 years ago	RPEPEGOLDENR	314	False	False	0.50000000 XCP

Inputs:

Time	Value (BTC)	Value (USD)	Address	#
2016-10-13 20:16	0.00147150	0.94	15DSGbbcWHV8Xy17E9ZyEGW2nKsdRWwmNU	0

Outputs:

#	Address	Value (USD)	Value (BTC)	Times
0	OP_RETURN d-7f4e736187e4d4279c5ee0ad24c85dd6	0.00	0.00000000	Unspent

OP_RETURN: j3��ay ��e�|:�jE�oA���'���iC������XG��q���0.wG��

10-13 23:43

Bitcoin nodes don't know what Rare Pepe is, but they can forget the OP_RETURN transaction.

High fees, block size wars, people went to other chains

Fair question: what is the right size for OP_RETURN?

Answer: make it cheaper than the alternatives, but not too cheap

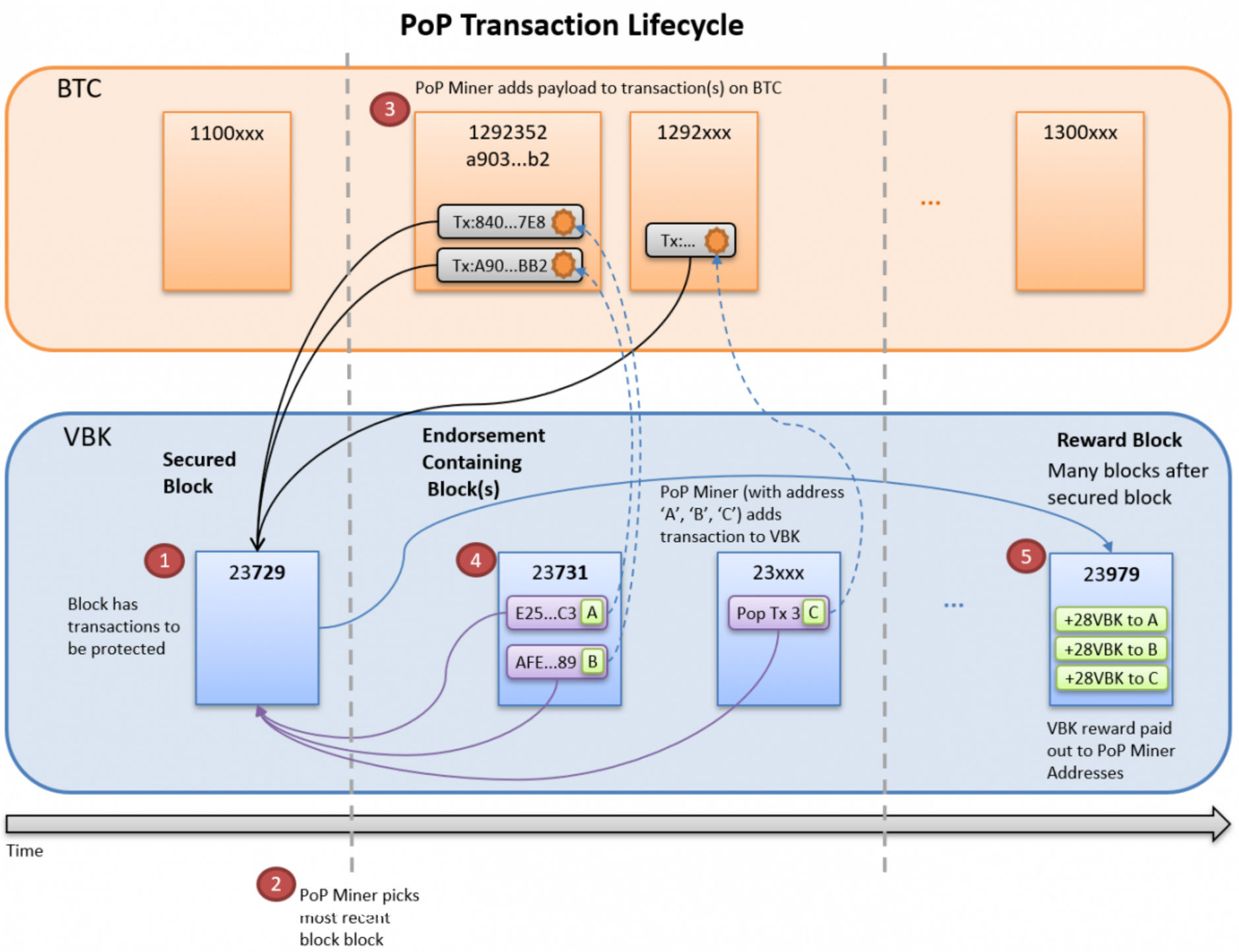
A New Blockchain Project Is Generating 20% Of Daily Bitcoin Transactions

Kyle Torpey Contributor ⓘ
Crypto & Blockchain
I've been a full-time Bitcoin writer and researcher since early 2014.

(Getty) GETTY

Recently, Bitcoin developers noticed some strange activity on the blockchain. Specifically, a large number of unidentified OP_RETURN transactions were discovered.

Winter is coming



takeaway: spam the Bitcoin blockchain, earn tokens

Equilibrium fee level?

Garzik wrote OP_RETURN code

Keep off the chain

E.g. RGB protocol

1. Client side validation
2. Pay to Contract

Indistinguishable from normal transaction, so can't be censored.

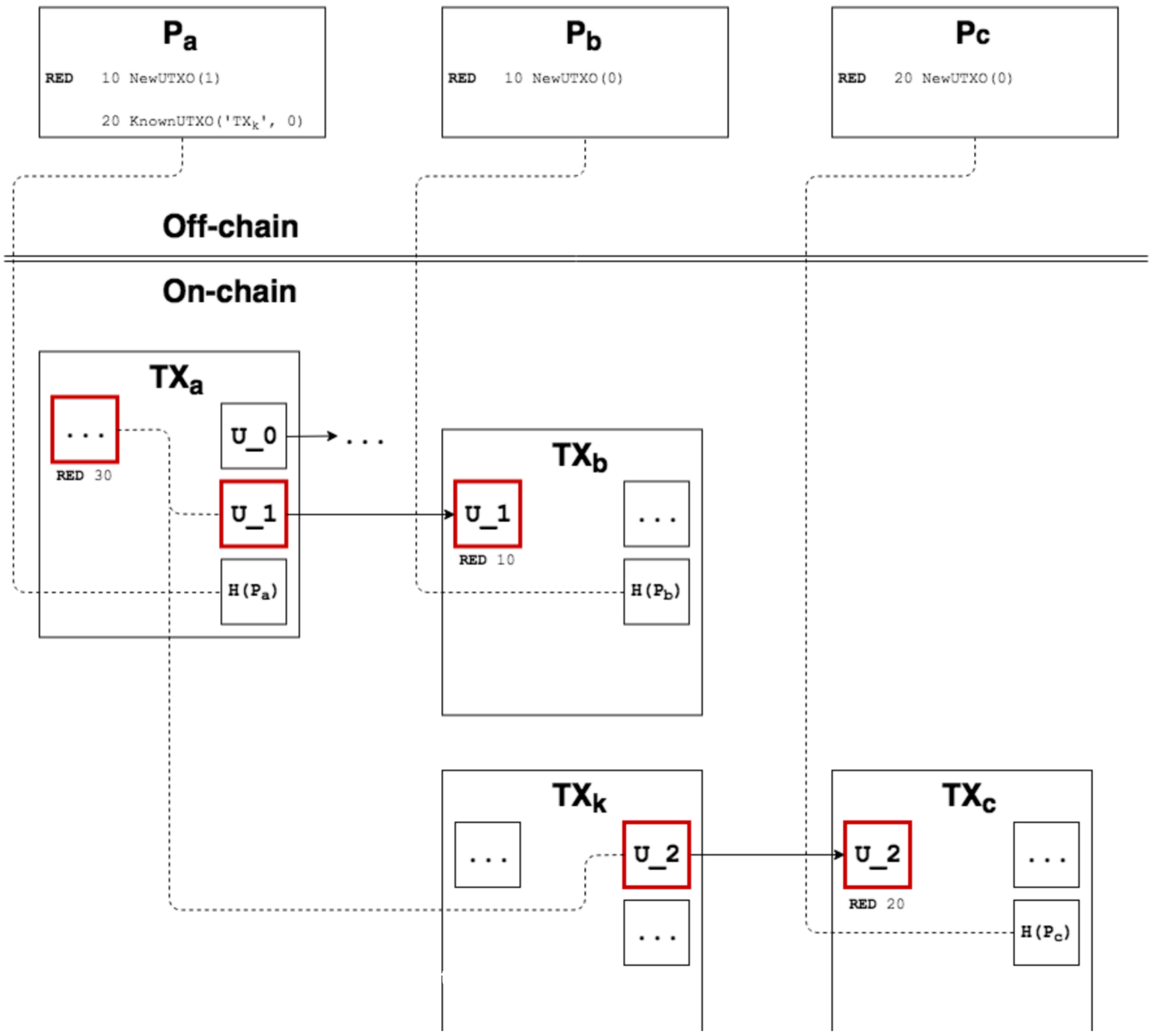
youtube.com

We Are Giving Away 10,000 Ethereum!!

Ethereum Giveaway!

— RGB asset: 100 finney fully backed by me
— Bitcoin testnet private key: cP7453tMvkWPcEHfx8zpCQezoU5PeoQjjEYaCMzxD9Tz4f5GEpxW (access to 10 finney)
— Instructions: https://github.com/Sjors/presentations (after my talk)

What makes RGB special: Client-Side-Validation

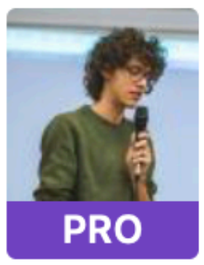


Confusing image, point is that on-chain txs are interpreted off-chain

Client-side Validation

```
kaleidoscope issueasset --title "ETH (Finney)" --supply 1000
Asset ID: 6f840761c0b7d0af3514c4577af80899b65a1bf2c7d022a6c0c58afa2f8f2bc9
```

```
22  pub struct Contract {
23      pub title: String,
24      pub issuance_utxo: OutPoint,
25      pub initial_owner_utxo: OutPoint,
26      pub burn_address: Address,
27      pub network: Network,
28      pub total_supply: u32,
29  }
30
31  impl Contract {
32      pub fn get_asset_id(&self) -> Sha256dHash {
33          self.bitcoin_hash()
34      }
35  }
```



Alekos Filini afilini

::1/128

Member of Developers Italia, BHB NETWORK, and 1 more

Current implementation by Alekos Filini (code sometimes more useful than spec)

Transaction on blockchain refers to asset id

asset id commits to rules, such as total supply and the initial owner

Client-side Validation

```
Asset ID: 6f840761c0b7d0af3514c4577af80899b65a1bf2c7d022a6c0c58afa2f8f2bc9
sha256(6f840761c0b7d0af3514c4577af80899b65a1bf2c7d022a6c0c58afa2f8f2bc9)
c92b8f2ffa8ac5c0a622d0c7f21b5ab69908f87a57c41435afd0b7c06107846f
```

Outputs / Receivers

Index	Address	Value (฿T)
0	tb1q468qs808t7jqfalgnjkyp8hacga3as5k3nvdjs	0.00997000
1	Nonstandard	0.00000000
		0.00997000

Input, Output Scripts

Index		Script
Input	0	Witness: 30440220233b0379348d9e85e9d29820b8ba59c912288866403dc7d118cbc6604df1cf2302203874373934fb8674aa032ab0d20c535d7fd407379d1a27f177af7fddd78ed5170103873888d8115beee6938510de7ddef0f1b22a2474e5ba58d4fccbde7766494bc6
Output	0	0 ae8e081de75fa404f7e89cac409efdc23b1ec296
Output	1	OP_RETURN c92b8f2ffa8ac5c0a622d0c7f21b5ab69908f87a57c41435afd0b7c06107846f **Decoded**: ?+?/?????"????????zW25????a?o

Client-side Validation

```
kaleidoscope sendtoaddress tb1qj7mtznsd6uzmztma6yutkklv4ypjj4g9mhmmf4 \
             6f840761c0b7d0af3514c4577af80899b65a1bf2c7d022a6c0c58afa2f8f2bc9 \
             100
```

Outputs / Receivers		
Index	**Address**	**Value (฿T)**
0	tb1qj7mtznsd6uzmztma6yutkklv4ypjj4g9mhmmf4	0.00497500
1	tb1q226hekesveps4tdnzdd4dvhc2dca05alr4axk8	0.00497500
2	Nonstandard	0.00000000
		0.00995000

Input, Output Scripts		
Index		**Script**
Input	**0**	Witness: 304402202e623b08767ac8ffbba1d126ac9bf69bccdbb4c17ed08e105c84f62e1b673b0202201e7bcf18a93448231d93417f509a7dedd7a742f4ec35f7d187fda2632735633101 0231f22d4601b26176c155099e8a9ec4285851c1a78249bd74f84b36eb4439b7b5
Output	**0**	0 97b6b14e0dd705b12f7dd138bb5beca903295505
Output	**1**	0 52b57cdb3066430aadb3135b56b2f85371d7d3bf
Output	**2**	OP_RETURN 31e5bf11b1118f30b7233c13aae207fddd75cf036e359762055b75b7feffea86

Client-side Validation

```
117  impl Verify for Proof {
118      fn get_needed_txs(&self) -> Vec<NeededTx> {
119          let mut ans = Vec::new();
120
121          for out_point in &self.bind_to {
122              ans.push(NeededTx::WhichSpendsOutPoint(out_point.clone()));
123          }
124
125          if self.is_root_proof() {
126              let mut needed_txs = self.contract.as_ref().unwrap().get_needed_txs();
127              ans.append(&mut needed_txs);
128          } else {
129              for i in 0..self.input.len() { // iterate the input proofs
130                  let mut needed_txs = self.input[i].get_needed_txs();
131                  ans.append(&mut needed_txs);
132              }
133          }
134
135          ans
136      }
```


Commits to chain of proofs

Proofs are uploaded to a server, identified by outpoint (UTXO + index) (or put in a zip file, like in my demo)

Pay to Contract

— OP_RETURN is suboptimal
— Tweak receive address to include hash
— good explainer [4]

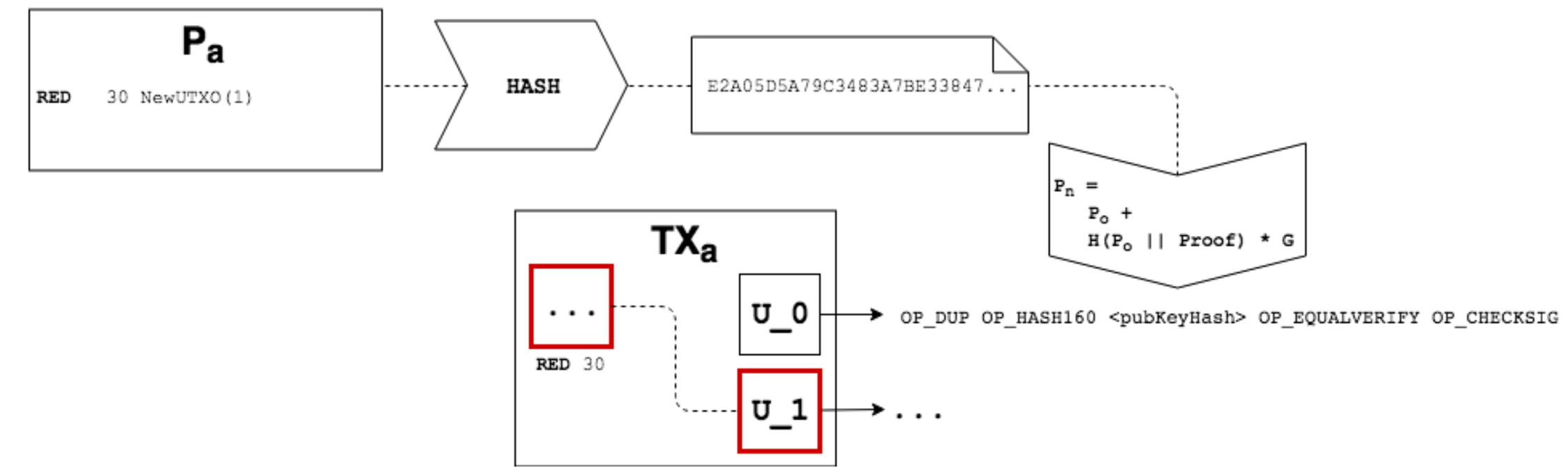
[4] https://blog.eternitywall.com/2018/04/13/sign-to-contract/

Similar mechanism used by Taproot to add secret spending conditions

In order to spend you must know the hash, so no accidental spending like with color coins

Recipient won't even see funds without this hash

Pay to Contract



Part of RGB spec, but not implemented yet

ETH Giveaway

Testnet private key: cP7453tMvkWPcEHfx8zpCQezoU5PeoQjjEYaCMzxD9Tz4f5GEpxW
Address: tb1qj7mtznsd6uzmztma6yutkklv4ypjj4g9mhmmf4

Asset ID: 6f840761c0b7d0af3514c4577af80899b65a1bf2c7d022a6c0c58afa2f8f2bc9

The above address contains 10 Finney worth of "stablecoin".

Redeem process

Install Bitcoin Core and launch with
`-testnet -server=1 -rpcuser=bitcoin -rpcpassword=bitcoin`.

Install Kaleidoscope and create `~/.rgb/rgb.conf`:

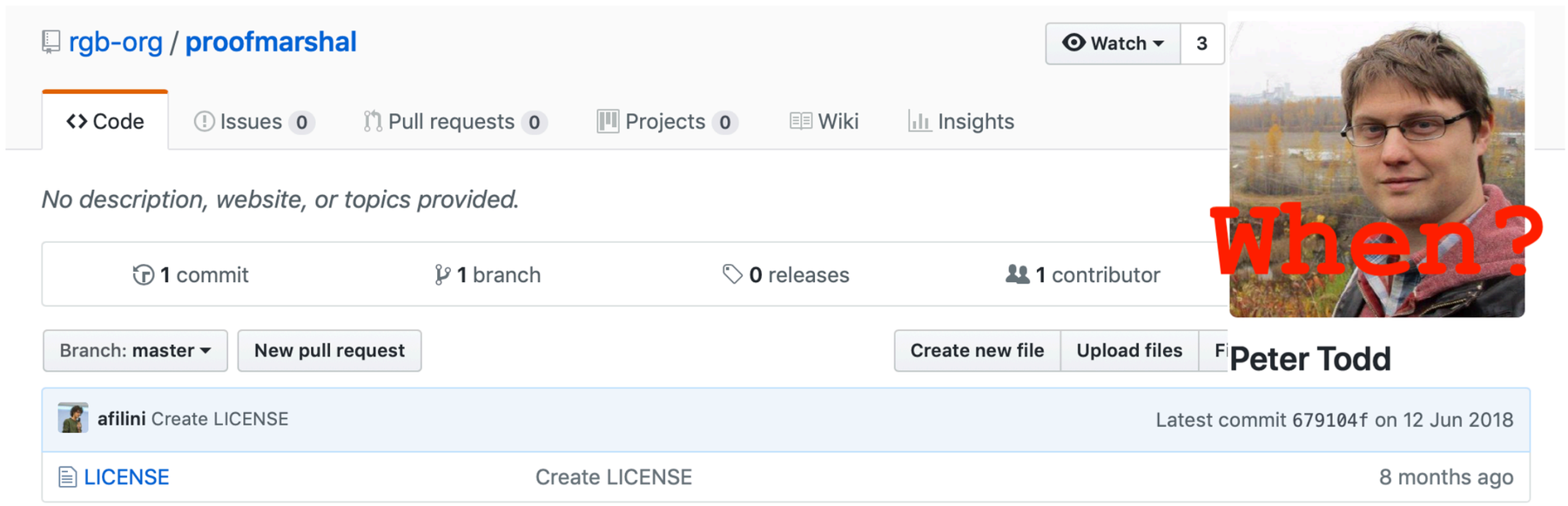
```
{
    "rpcconnect": "127.0.0.1",
    "rpcport": 18332,
    "rpcuser": "bitcoin",
    "rpcpassword": "bitcoin",
    "default_server": "localhost:3000"
}
```

Install Bitfröst, create `~/.rgb-server` and extract `rgb-server.zip`, which contains the proofs you need in order to spend the stablecoin.

Send up to 10 finney to me at
`tb1q4fnez58x6nqrjvyx7hrp8g92wc9ne439jag2ns@localhost:3000` and email the generated proof to `sjors@sprovoost.nl` as well as your ETH address.

Scaling - Proofmarshal

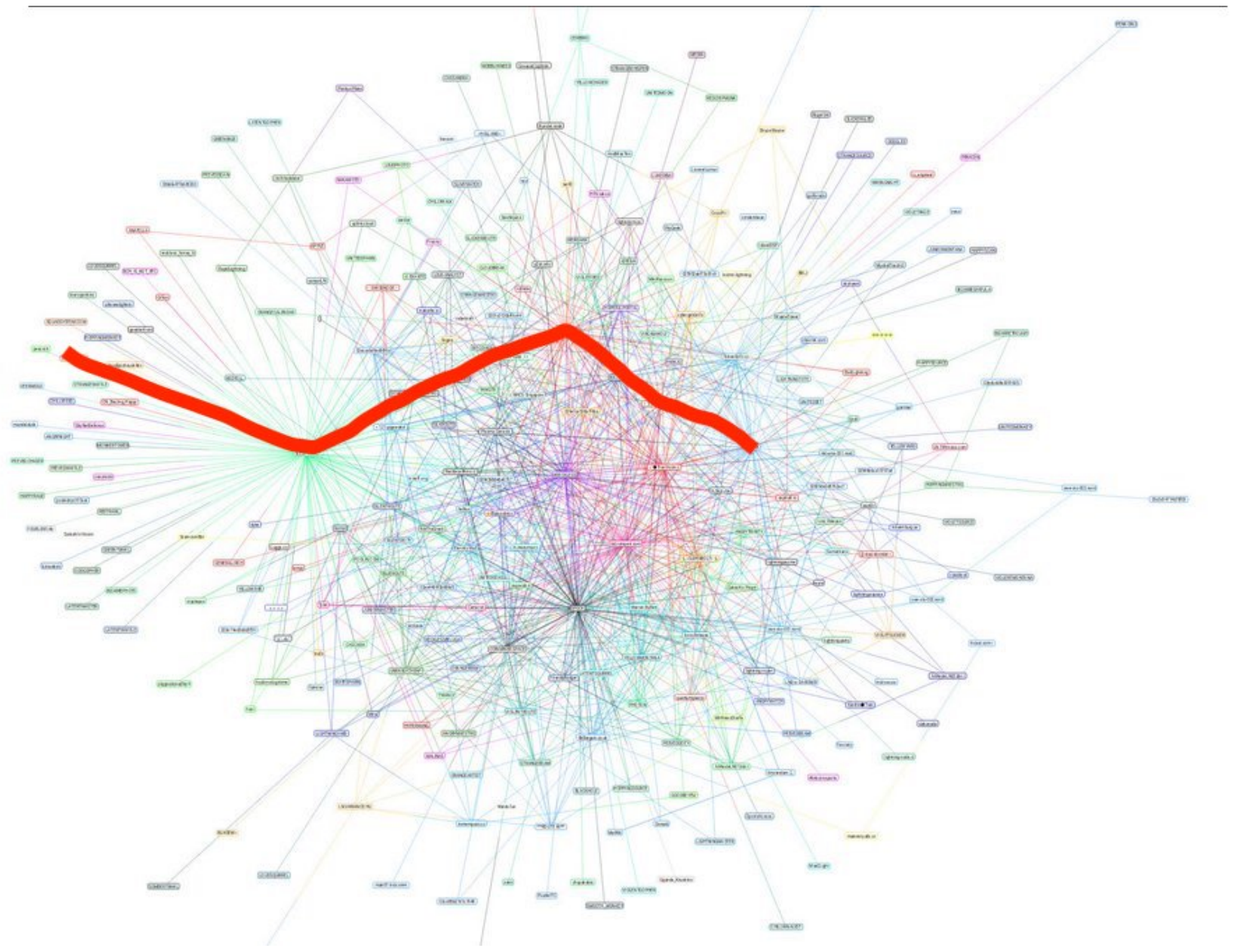
— Multiple transactions in one UTXO using a Merkle tree



Don't yet understand how this works, figure out during Q&A?

Gist afaik: instead of single UTXO for each move, use Merkle tree to compress multiple proofs into one UTXO

Scaling - Lightning (colored channels)



Colored channels

— Study RGB: https://github.com/rgb-org/spec
— Help improve it before the hackathon (implementing pay-to-contract)
— Don't like it? Make something else.
— ETH Giveaway instructions: https://github.com/Sjors/presentations

PGP: ED9B DF7A D6A5 5E23 2E84 5242 57FF 9BDB CC30 1009