Guld: Universal Protocol for Relative Consensus

Author: Ira MIller

Path: life/isysd

Email: public@iramiller.com

Abstract

A polytree with signed, typed, symmetrical nodes can be used to represent, record, compare, and perform operations on individual and group perspectives on any topic describable in a digitized record. This `blocktree` would then represent a virtual universe for the participants, including natural causality and one-directional "time". The participants would be able to make assertions about each other's perspectives, building proofs of consensus or dissent. Using existing technologies like git, pgp, and ssh such a polytree can be constructed in a timely and accessible manner, while maintaining scalability and decentralization at the protocol level.

Introduction

Since Bitcoin(1) introduced the blockchain concept in 2009, hundreds of experimental technologies have been developed to help users achieve consensus on one topic or another. The majority of these have been organized around the model of a central, unbranching trunk of absolute truth, i.e. a blockchain. To "fork" a blockchain, is perceived as negative, because it breaks the implicit or explicit contract of absolute truth that formed the chain.

Perhaps due to their unforkability, blockchains have proved inflexible at managing conflict and change. Though Turing complete contract systems have been created(2), these have so far been insufficient to address the problem of the Firm(3), let alone the Government(4). This restricts blockchains to the realm of prices and payment systems, and outside of the firm.(5)

A relativistic blocktree of perspectives would be a much more flexible and powerful platform for propagation of information in contracts, communities, firms, and governments. Such trees operate on abstract beliefs and observations to create causal relationships between nodes, and allow efficient proofs.(6)

Individual Observers

"I think, therefore I am"(7) is a beautiful and strong proof, but it only works for the observer himself. A modernization using asymmetric cryptography would be "I sign a unique perspective using this key, therefore I am."

To show the distinction, lets consider the case of Rene Descartes, the author of the original quote. Was he real, or is he a figment of your/my dreams? Even he would admit that we have insufficient evidence to prove the former. But consider, what if he had generated an asymmetrical keypair using an algorithm such as RSA, not invented until 1983.(8) Suppose he had somehow signed his book, publishing it with his public key in an appendix. Would he not then be able to provide a signature upon request proving that he is the same person that observed the original proof? Suppose he had somehow passed the private key down through the generations of his family to today, without it's secret being revealed publicly. Would not his heir be able to provide proof to us, even today of the relationship between the key and the work? This inherited key perspective would not be ideal, but it would be sufficient proof even

for M. Descartes to accept.

Is the key alone sufficient proof to accept a new `life`? Surely not, because the key is an inanimate math phenomenon that exists in nature. A thought which provably could not have come from yourself must, however, be evidence of some other thinker. Multiple thoughts signed by that same key indicate a strong pattern that the signer is the thinker. Daily use of the key in the pattern of the thinker's life make it almost certain.

So, let us define a `life` (in `blocktree` terminology) as an entity capable of generating an asymmetrical keypair, and using it to publish a unique perspective. This proof is still relative, but now it is relative to a second party, the observer of the thinker. The observer then needs to ask themself if the keypair and perspective are suitably unique. This process can then build, by the observer publishing their perspective on the original thinker, each able to make provable assertions about the other's perspective, and the state of their network.(5)

Proof of Labor (Observation)

1. RD signs & publishes unique thought A, along with public key RDKey
2. Observer validates signature of RDKey.
3. Observer decides on uniqueness of thought A as proof of a living perspective.
4. If Observer accepts thought A as unique, Observer can infer that the signer RD using key RDKey is also alive.

Perspectives (tree nodes)

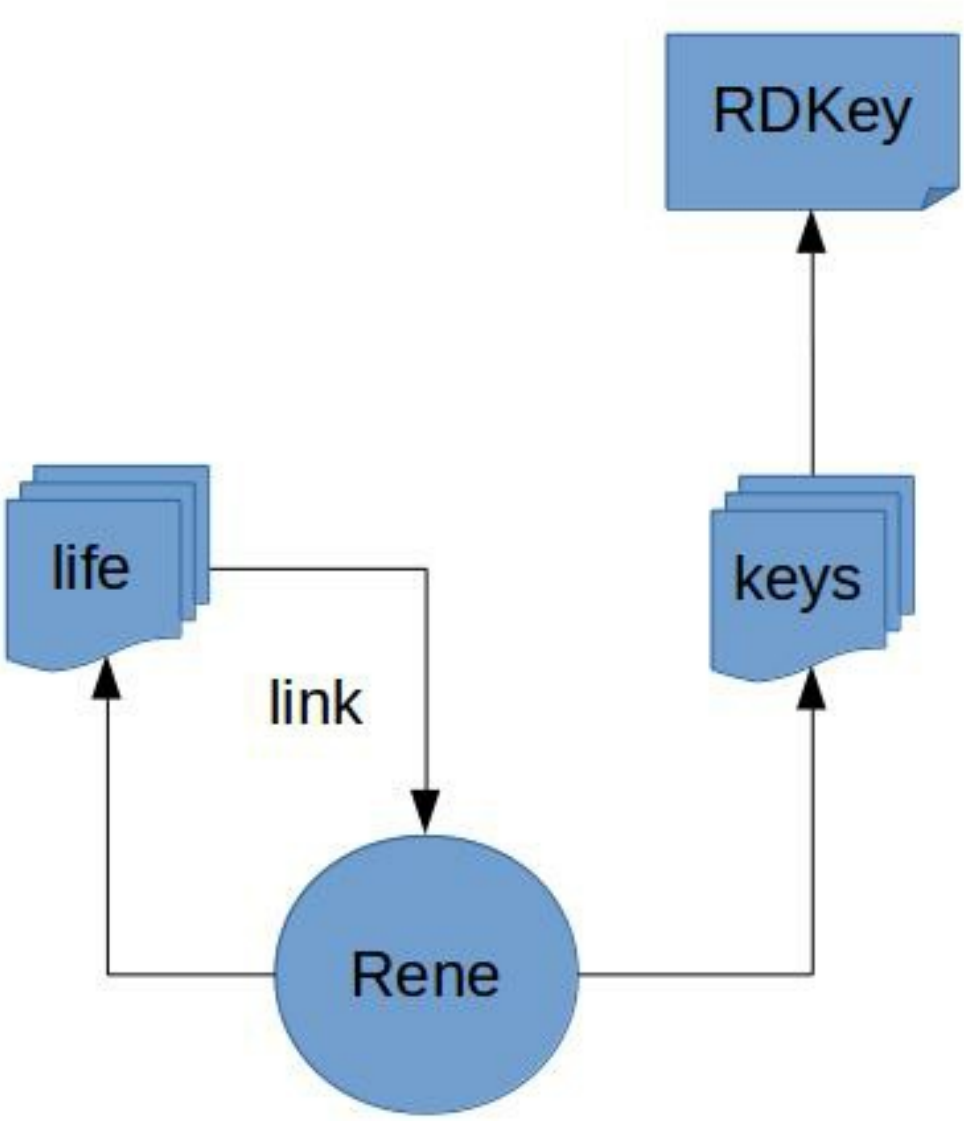
The blocktree needs a data type for any observer's perspective, which make up primary nodes in the polytree. In practice, a plain text data type like JSON should be used, but abstractly, the perspective can be thought of as a dialectic logic tree. That is, each node inside the perspective asks

a boolean question.

The simplest and smallest perspective would be Descartes's "I think, therefore I am", where his thought is his public key.

path	question
`life`	Is the record a `life` as defined by PoL?
`life/Rene`	Is the record the signer known as `Rene`, who uses key `RDKey`?
`keys`	Is the record an ascii armored PGP public key?
`keys/RDKey`	Is the record the key known as RDKey?

The entries are either the raw contents that answer the question, or a git submodule linking to said contents. In the case of category nodes, like `keys`, the raw contents may be a directory, which in git, is just a path to a hashable object.

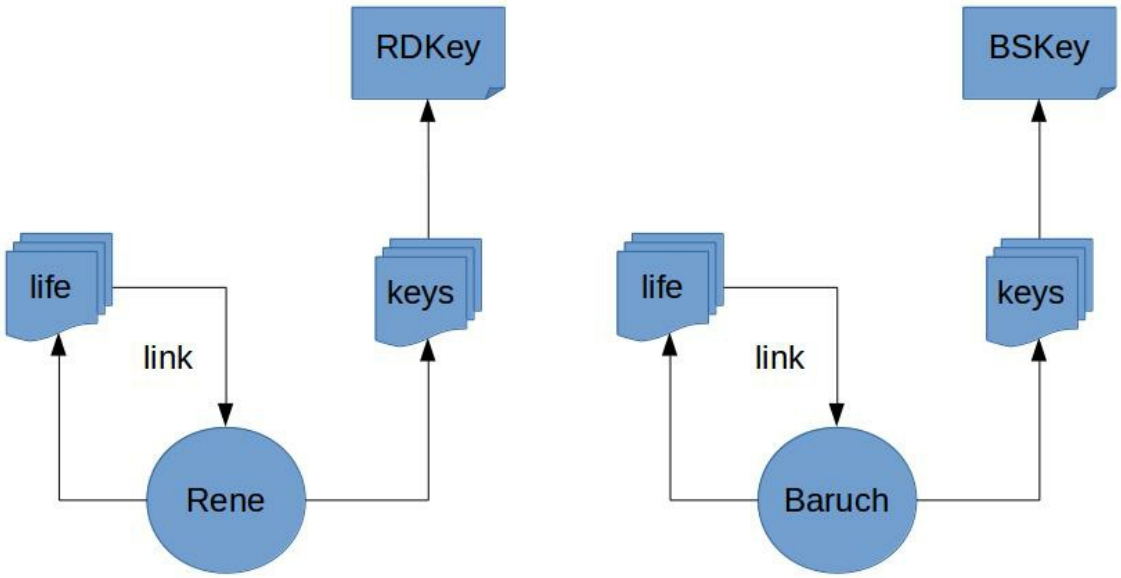


The 'life/Rene' case is more complex. What is Rene? In the git tree, it is a submodule, referencing the HEAD commit of the parent perspective. More subtle, though, Rene is the pattern of observations that are always signed using `RDKey`. To help us recognize this, Rene keeps all of these observations in the agreed upon perspective taxonomy, at his dedicated namespace in the tree, `life/Rene`.

Since observers recognize each other, the blocktree is symmetrical, and potentially contains an infinite number of perspective branches. Though observers can use different formats for their perspectives, this disrupts

the functionality of the blocktree, as relationships only develop through provable consensus, i.e. symmetry. So Rene must use the conventional perspective taxonomy, or certainly miscommunicate.

Assume there is another rational entity Baruch Spinoza, who performs a similar self-proof using BSKey. The two would not necessarily see each other at first, and would be in self-contained perspective loops. The perspectives would be symmetrical, and easily comparable, however, should any outside observer read them both.

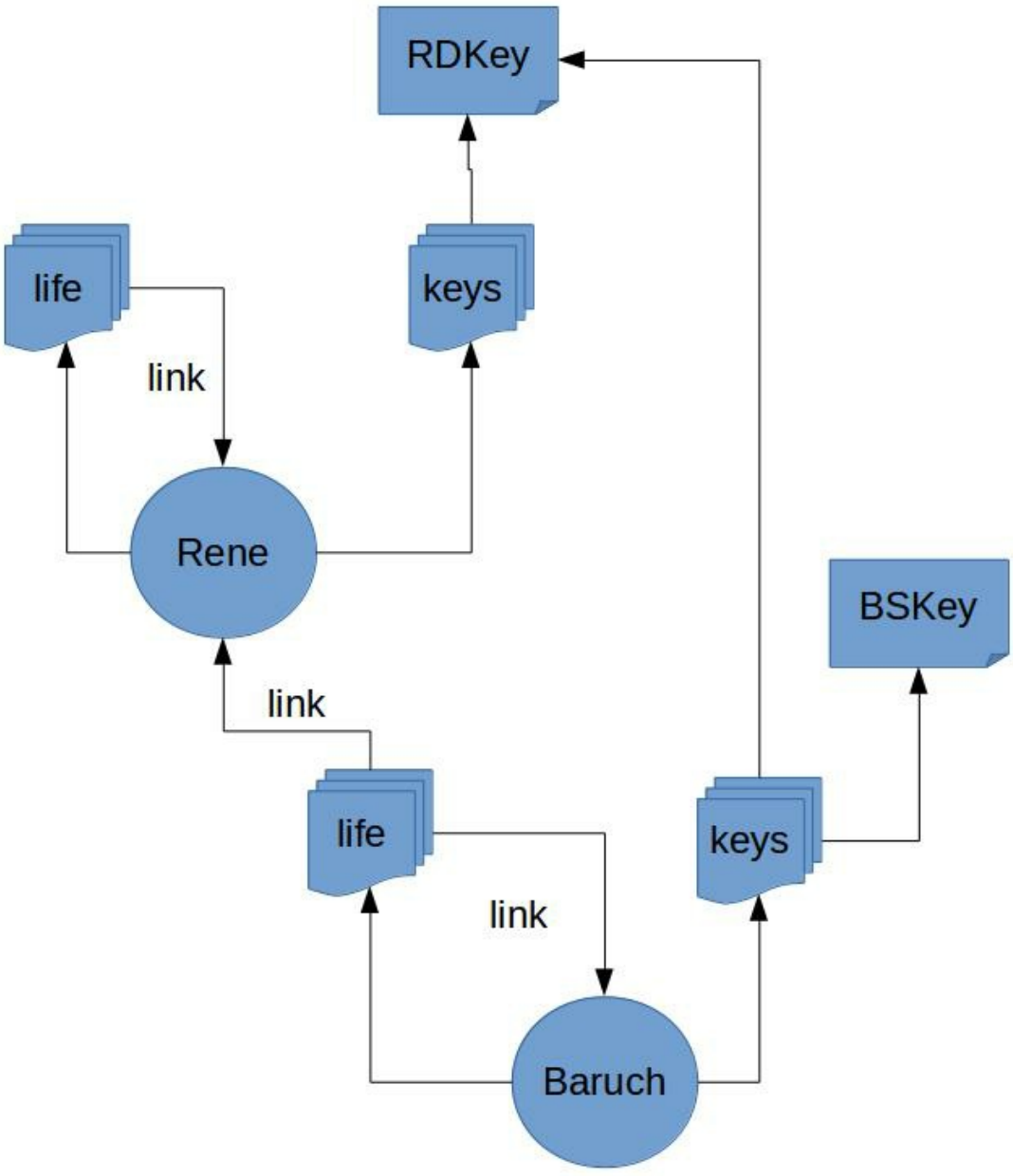


The Blocktree

The blocktree itself is a combination of many perspective nodes. In the guld software, each perspective node will be a git repository, and the links between perspective nodes are git submodules.(9) Since submodules are not the complete contents, but rather a SHA1 hash, this maintains both the privacy of the hashee and the disk space of the hasher.

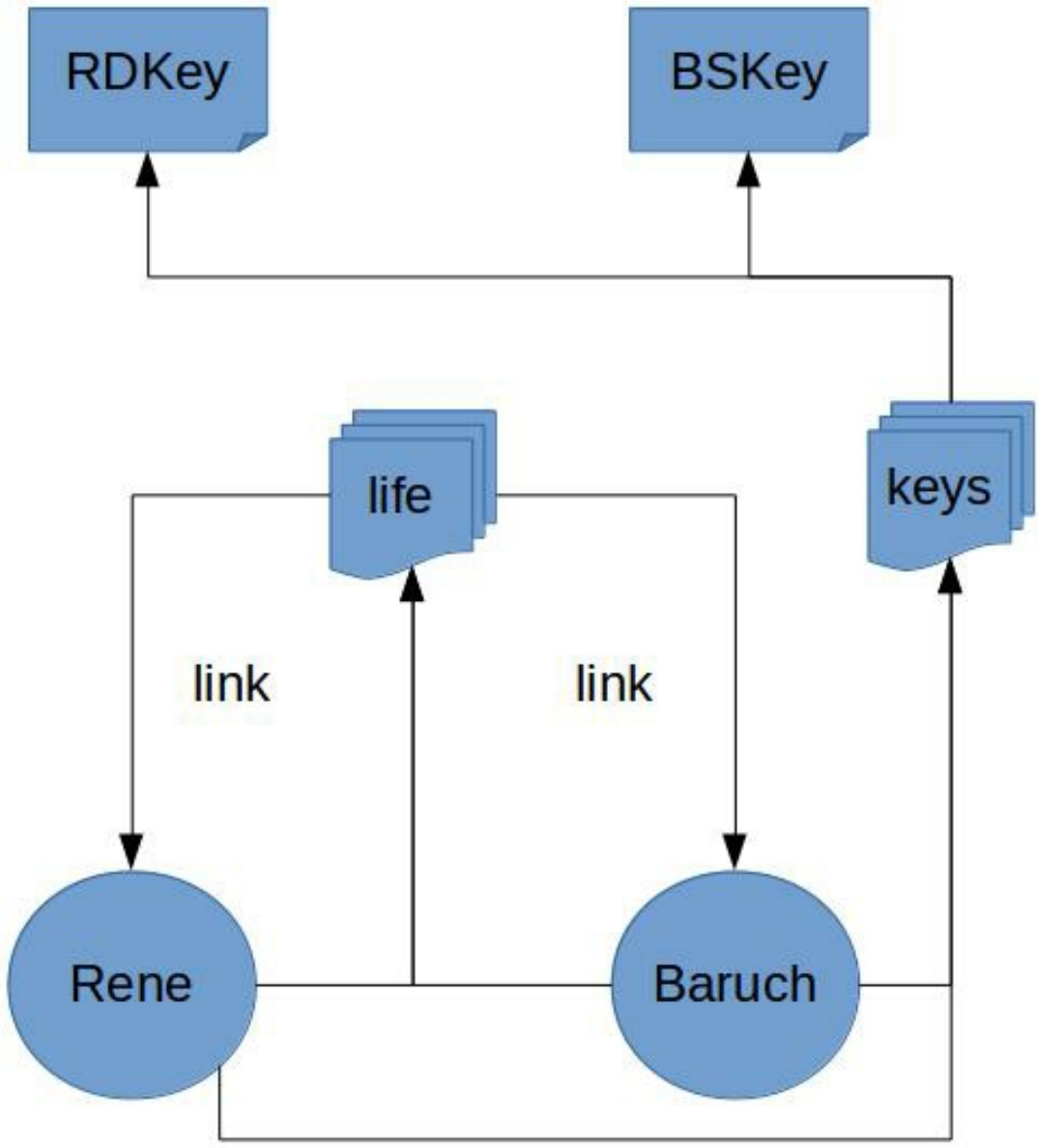
Baruch could read Rene's theory, and observe that RDKey was still in active use. Baruch would therefore update his tree by referencing Rene,

and RDKey in their respective places. This would yield a self-similar, predictable and organized tree. Baruch could make basic assertions about Rene's perspective, such as "Rene believes he exists." Rene would not necessarily accept any premise of Baruch's, but would be able to prove "Baruch believes Rene exists".



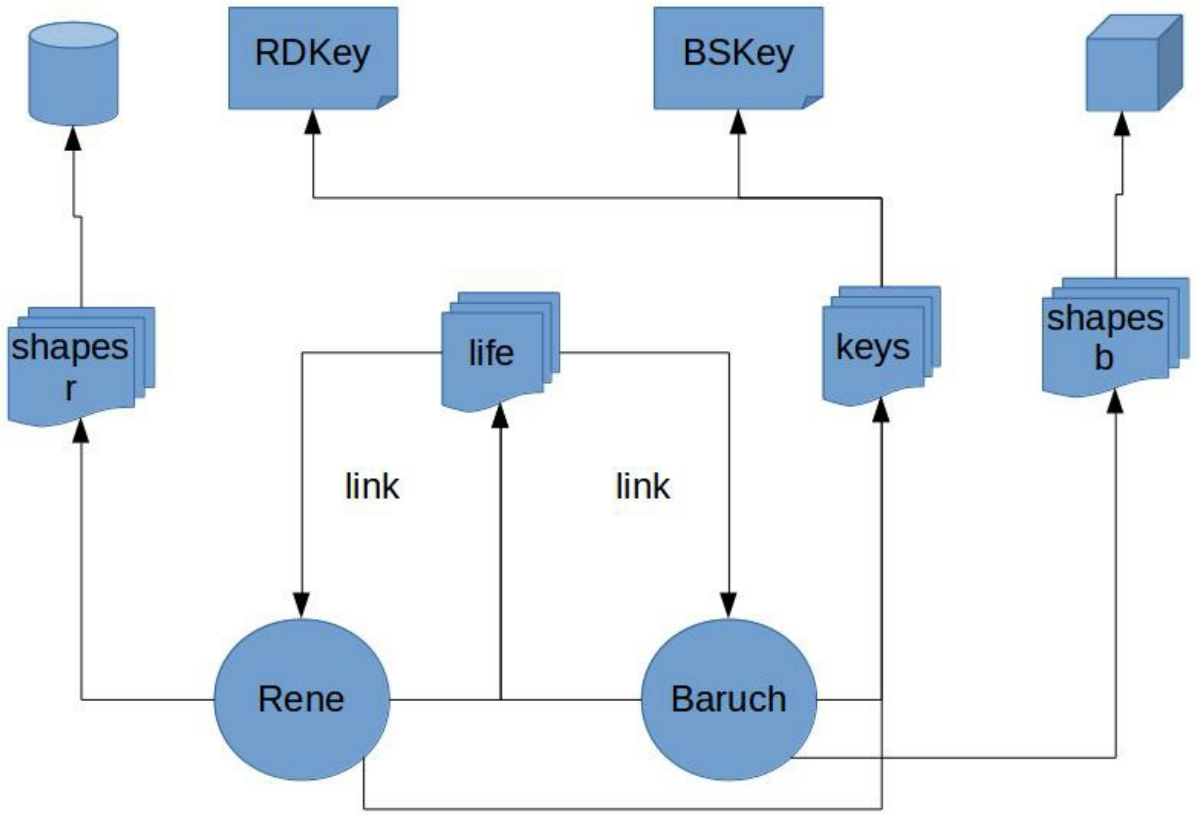
Next, let's assume there is a two way communication between Rene and Baruch. Rene sees that Baruch has recognized his work, and reciprocates, also yielding identical hashes for `life`, and `keys`. The two have achieved consensus.

The simplest state of consensus is a symmetric equilibrium of proven existence between two parties. That is, both parties have shared public keys, and proven to each other their sentience with a unique thought. In git practice, an additional step of merging trees may be necessary, but that's procedural.



When in a state of absolute consensus, as shown above, this tree is one level of perspective nodes deep, but n rows wide, where n is the number of unique observers. In git terms, every submodule in the `life` directory would point back to the parent repository.

When there is disagreement, each `life` node may represent one or more branch points. The topography of this branching creates a control surface to be manipulated. For instance, suppose the branching is on the topic of 3d shapes. Their trees would overlap in some areas, differ in the area of shapes, and each contain at least a reference to the other.

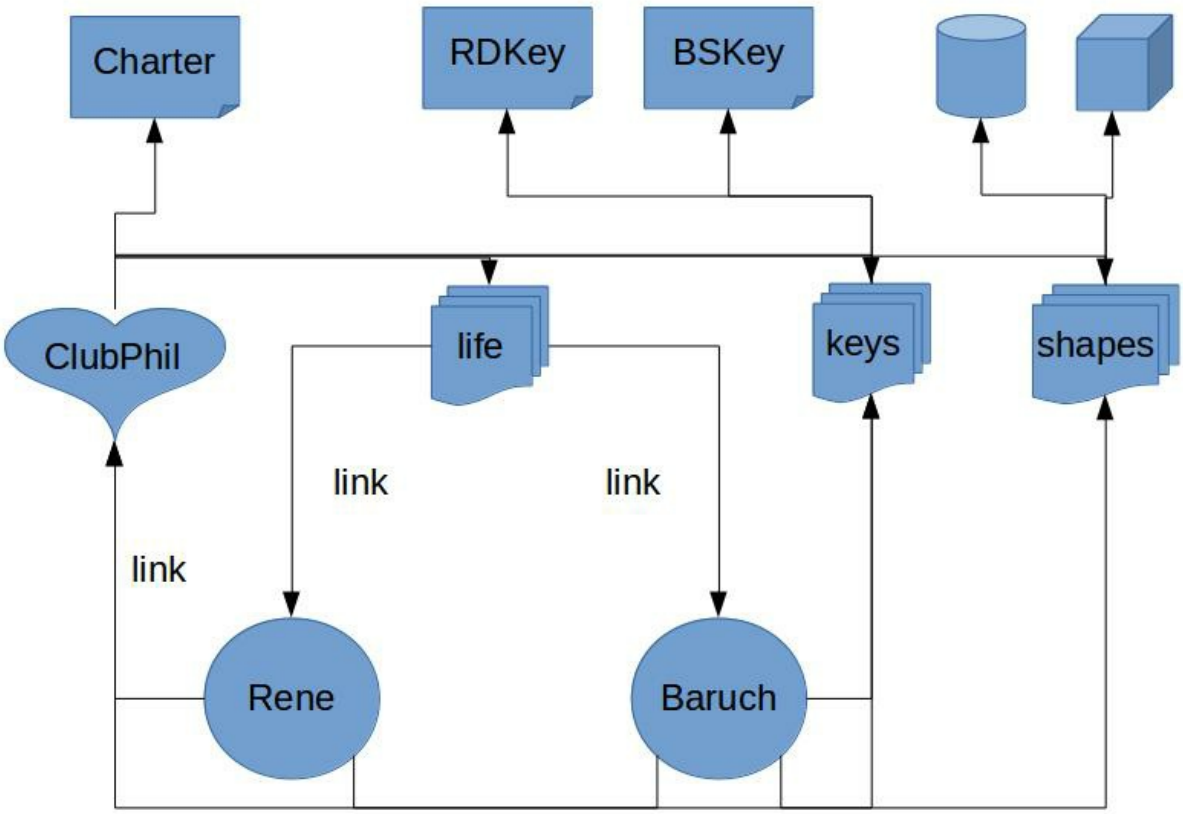


Though Rene does not himself recognize the `cube` as a shape, he would be able to reference it using Baruch's definition, and vice versa for Baruch and the `cylinder`.

Communities

Suppose that Rene and Baruch talk it out, and come to consensus on the subject of shapes. They're so excited about finally reaching accord, that they decide to form a philosopher's club based on the subject, `ClubPhil`.

Suppose `ClubPhil` is based on a mutually exclusive charter that said "all members must always agree on the subject of shapes". Suppose furthermore, that the members of `ClubPhil` are charged with keeping an official perspective for the community, including any unanimously recognized shapes, keys, and lives aka members.



Each member of the community would be able to prove at any time the perspective of any other on any record, including the contractual shapes, and the charter. If any member was to recognize a foreign shape, for instance, the others would be able to prove breach of the `ClubPhil` charter.

Then let `community` in the `blocktree` be defined as an agreement between mutually recognized `life` s to publish a combined perspective.

Co-signing aka Voting

How then would the `ClubPhil` community progress? What would happen if one day Rene discovered the `bevel`? Based on their charter, another argument, followed by a community fork or a unanimous update. A more sophisticated contract could be written with some effort, that would take into account the process of debate, creating a state machine for the community. Debates would take place on member branches, and only be merged into the community branch upon reaching a threshold of approval.

Because the community's state will be hashed into each user's perspective at each event the user observes, each user creates a constantly affirmed feed of their perspective on the state of the community and all relevant records. Since each record is behind a `git` submodule, it only takes up the space of a hash. Furthermore, since `git` only tracks changes, only the changed references need be considered or stored in each commit block.

Therefore, counting "votes" on a community issue is trivial. Simply run the "x believes y" proof for each member x on the topic y. If the pre-arranged community threshold (i.e. 50%, 100%) for merging has been met, every member knows to merge topic y into the community consensus branch.

Different weights can be given to member votes within a community, or even special roles and responsibilities. Since the basics of counting and tracking states do not change, these decisions will be left up to community developers.

Finance

The typical blockchain functionality of sequential, trustless transactions can be achieved by filing a ledger in a community branch of the blocktree. The community can manage the state of the ledger, in the same way that

`ClubPhil` manages the shapes set.

While such a ledger could theoretically use any format, plain text is most user friendly, and efficient in the chosen file system of git. It is therefore recommended to use `ledger-cli` for all blocktree ledgers.(10)

The `ledger-cli` format has very powerful unit control, including support completely custom commodity strings. Combined with the consensus a community can provide, this allows new digital tokens to be issued and controlled.

```
2017/06/7 ball4thegame
  Assets:ball4thegame:Ambassador          1000 XGC
  Equity:guld:Ambassador                -1000 XGC
```

Network

Though the standard `git` software package ships with a server, and that server supports p2p `ssh` authentication, this is rarely made use of. In practice, users tend to use one of the popular hosting services, such as github, bitbucket, gitlab, etc.. One of these is more than sufficient for the average open source project, but not for a consensus network. Thankfully, git supports a multiple `remote` hosts for each repository, and the protocol strengthens the user's signed ownership of the state, making each inter-changeable.

```
$ git remote -v
isysd    git@guld.host:isysd/guld.git (fetch)
isysd    git@guld.host:isysd/guld.git (push)
isysd-github    git@github.com:isysd/guld.git (fetch)
isysd-github    git@github.com:isysd/guld.git (push)
isysd-bitbucket    git@bitbucket.org:isysd/guld.git (fetch)
isysd-bitbucket    git@bitbucket.org:isysd/guld.git (push)
```

Because we recognize that `isysd` is a living person who uses a specific PGP key to publish to one or all of the mirrors shown, we can accept the most up to date with his signed commits. The hosts are not to be trusted. Logistically, checking all of these is inefficient, so users should establish P2P socket connections with their friends to send notifications about commits and hosts.

Any mutually accepted users of the protocol could publish encrypted contact info for each other, including IP addresses for git hosts and live socket sessions. Ideally, each users on the network would host their own `git` servers, using software like gitolite(11) to manage permissions in repository, and publishing their IP address (selectively encrypted) all on the blocktree.

Example Gitolite configuration

```
@users = u1 u2 u3

repo foo/CREATOR/[a-z]..*
    C   =   u1 u2 u3
    RW+ =   CREATOR
    RW  =   WRITERS
    R   =   READERS
```

This would create a provable hosting service, where the host can demonstrate compliance with community access rules, and even put the configuration itself in the control of one or more signing users.

Running `git` and `ssh` servers is an, as yet, under-assessed security risk to request of end users. Until there is are lots of data from experienced systems administrators, and the protocol has stabilized, end users should not be required to host their own repositories.

Starting with professional, trustless hosting services, the guld network will evolve to proven hosts, finally to a completely p2p, self-hosted network.

This evolution will separate guld more and more from traditional infrastructure like Dynamic Name Servers (DNS), gradually reshaping the internet on P2P terms, with P2P identities, governance, and finances built in.

Conclusion

The `blocktree` represents a rich, self-contained world, with causality and enforceable laws or it's participants. By using relative proofs based on perspectives, relative truths and consensus can be reached, lost, and reachieved flexibly, in ways that blockchains are unable to replicate. This blocktree of relative truth, or truth good enough for a community, is superior to blockchains of absolute truth for human contracts, firms, and governments.

References

- 1 Satoshi Nakamoto "Bitcoin" (2009)
- 2 Vitalik Buterin et al "Ethereum" (2014)
- 3 Christoph Jentzsch "The History of The DAO and Lessons Learned"
- 4 The block size controversy entyr on the bitcoin wiki.
- [5] R. H. COASE "The Nature of the Firm" (1937)
- 6 Jin H. Kim and Judea Pearl "A COMPUTATIONAL MODEL FOR CAUSAL AND DIAGNOSTIC REASONING IN INFERENCE

SYSTEMS"

- [7] Rene Descartes "A Discourse on Method" (1637) translated by John Veitch (2008)
- [8] Rivest, Shamir, Adleman "Cryptographic communications system and method" (1983)
- [9] Linus Torvalds, Junio Hamano et all "git" (2005-)
- 10 John Wiegley et all "Ledger-cli" (2003-)
- 11 Sitaram Chamarty "gitolite" (2012-)