Revolution Populi

How to return digital power back to the people to whom it belongs

Table Of Contents

Objective

To return data sovereignty back to the people.

Structure

Revolution Populi (RevPop) proposes a 3-legged stool approach to advance this objective:

1) A decentralized user-controlled layer-1 blockchain database which allows any person to be accessed -- but only if they choose to be -- operated and maintained decentrally by a Decentralized Autonomous Organization (DAO), apart from any central authority, can be a sustainable core for a data sovereignty ecosystem.

2) An open source project to package up social net components, each compatible with the layer-1, and each made openly available, could allow dozens or hundreds or even thousands of social nets to bloom; in effect bringing to bear the ultimate "Facebook Killer Kit", and placing data tyranny on a plausible path to extinction, engendering a market for fair economic equilibrium between social nets and users.

3) A cryptocurrency clearing house mechanism / app built on top of the layer-1 can bring structural stability to the entire crypto trading marketplace, ushering in a new era of institutional participation by way of risk management backstops and structural confidence, and can serve as a stability beam to buttress and bolster an ongoing data sovereignty ecosystem.

Social Net DApps

Packaging social net components that're compatible with the layer-1 can be a powerful consumer app mechanism for both development and ultimately consumer scale. This type of framework would allow users to establish their account on the blockchain, and to therefore have the ultimate mechanism to control their data, and the value generated from it. It would allow a user, any user, to port their data between social nets (or any apps) as they see fit, to clip the apps that do things they don't like, to re-engage with them if they feel like it, and to be placed in a position of sovereign control over any value that's generated from their data (that's generated from their very "being"); and it would allow the elegant capability to utilize this value to directly consume any manner of products and services that may be offered within the ecosystem, all by way of a native cryptographic token.

Distributed Crypto Clearing

RevPop's decentralized DAO layer-1 blockchain would allow for atomic counterparty-risk novation when it comes to crypto trading — a digital risk-offloader for counterparties in crypto trades — ushering in a multilateral market where anyone could trade any crypto on a guaranteed basis (and not just ERC-20 tokens and a bunch of synthetics) without fear of the other side re-neg-ing. The decentralized RevPop layer-1 is a natural neutral base layer upon which an app / service could atomically clear crypto trades, with true counterparty risk novation through a clearing house structure. These mechanics can then be underwritten by a decentralized guarantee fund, where depositors earn yield once a trade settles, and they can earn it in RVP.

Token Use

The RVP token has multiple uses. Within the layer-1 ecosystem, a variety of apps can provide consumer services. The RVP token would be used for these transactions, for commercial purposes. As an example, several different social networks can compete with one another. The nature of business competition comes down to the quality of your product or service, and the price you charge for it. In social networks, the business model is about advertising dollars, which all goes to them. In this new ecosystem, since the user owns and controls their data, different apps with different feature sets (some more appealing than others) can compete based on the quality of their services, which has always been the case; but now they will need to compete on price too. Advertising revenue would need to be shared with users based on a competitive equilibrium. The price component is no longer what an advertiser is willing to pay a *platform* -- it's what an advertiser is willing to pay the *users* of the platform. From there, the user would get to decide what amount they're willing to share with the platform (if any), and not the other way around. Some social nets will be able to command x amount of these fees; others will be able to get y, and so on, based on the quality of their product. The advertiser then could pay the user directly for their portion, and the platform directly for theirs.

Additionally, since any app will be able to utilize this database, other services such as music streaming, could be brought to market in the ecosystem. And since all the data would be portable, including the data associated with token transfers, a music streaming service, for example, could be paid with the same token that a user makes from their social media activity.

Within financial services, interestingly enough, this layer-1 database is also uniquely designed to manage trade settlements, trade matching, and hosts of back office functions, especially in the cryptocurrency sphere. As services such as exchanges or clearing houses wish to use the layer-1 database, they too would be paid for their services in RVP in a seamless and atomically settled way; and products like the aforementioned DeFi guarantee fund could also use the RVP token.

Governance & Maintenance Use

By virtue of the rdPOS consensus algorithm, which is a proof of stake algorithm, blockchain witnesses "stake" in order to verify a block. These stakers then get paid in yield. Since it is a *delegated* proof of stake algorithm, and witnesses are elected by popular vote, so the RVP token also functions as a "maintenance" token. And since the DAO foundation would be responsible for issuing grants, also based on popular votes of token holders, the token also acts in a governance capacity for the direction of the overarching protocol.

Layer-1 Blockchain Topology

Summary

We propose a unique and elegant roundtrip structure for a blockchain architecture which is both groundbreaking and simple.

Social net (or any app) → server (e.g. EC2 or a decentralized server, it'll be up to the app) → API → RevPop layer-1 → pointer → storage → API → server → social net (or any DApp).

We've devised a method that can combine decentralization, security and speed. This layer-1 is a database enabling users to have control and ownership over their data, with other elements of the infrastructure residing off-chain to enable fast throughput. This, combined with an innovation on the Proof Of Stake consensus mechanism, would allow this layer-1 to service users at scale.

Access Controls

Users would be able to provide access to individual apps (any DApp) as they choose. The access provisions and other data and content can be recorded on the layer-1 blockchain. Each app accessing the blockchain database is responsible for choosing their own method of serving data. And data could be fetched only if the user allows access. Apps would pay the DAO fees, much in the way they'd otherwise pay a Software as a Service (SaaS) database provider.

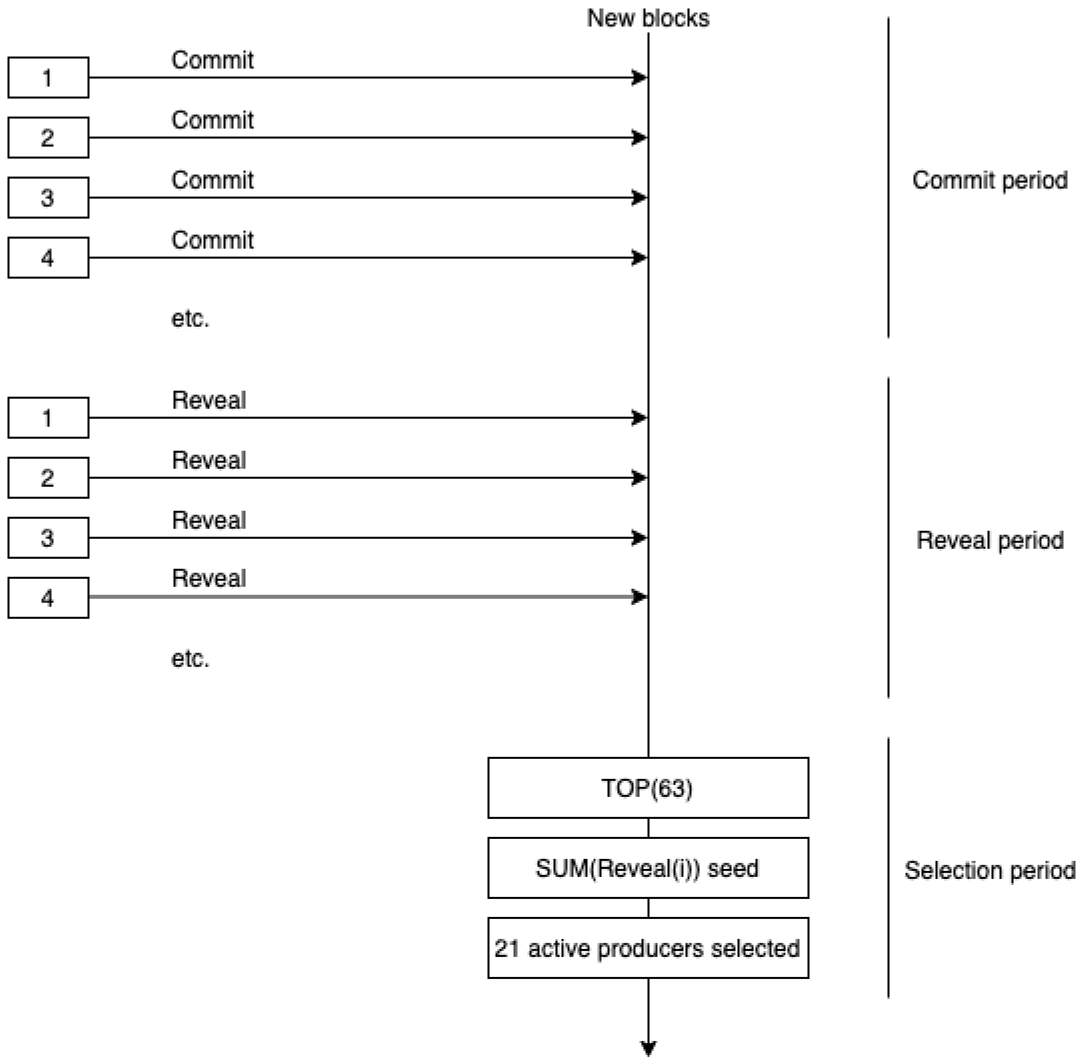
Consensus Mechanism

A protocol based on dPOS "delegated Proof Of Stake" can achieve a robust transactions per second (TPS) rate - using either dPOS itself, or something accretively innovative such as our plan for rdPOS "random delegated Proof Of Stake". A consensus mechanism with 63 pre-elected nodes, but with 21 randomly selected pre-determined nodes serving at any given time, can mean fast transaction speeds along with security, scalability and elected accountability. Due to the maturation of consensus algorithms, along with ongoing innovation and technological improvements, a modern social network, or any modern application for that matter, should have no problem operating on a blockchain technology foundation that functions as a database.

rdPOS behaves like its predecessor. At the final stage, during the vote counting phase, 21 active witnesses are chosen randomly from the list of TOP 63 candidates. This attracts more nodes to the production of blocks while keeping the number of active producers small, speeding up the blockchain. Test Net available at testnet.revolutionpopuli.com.

Pseudo-random number generation in the blockchain is not an easy task. We've implemented a commit-reveal scheme for obtaining a random seed each time 21 new active witnesses are selected. Now, by randomizing the selected block producers, coupled with regular, robust and public elections, the mix of producers is different for every block, and each producer remains accountable. This functionally removes the threat of "centralization" that had been at the heart of the critique of dPOS consensus in the past. It's important to note that the number of top elected candidates can be calibrated, e.g. if the system proves to be so lightening fast, and updating from 63 TOP to 126 TOP has de minimis (or acceptable) impact on speed, then such a change can always be implemented if the community deems it worthwhile - or the community can even seek to randomize the total number of TOP witnesses per cycle. The system, by the nature of its structure can be self-improving and ever-striving to functionally obviate the "Tri-lema", to allow speed, security and decentralization to all co-exist in harmony.

In short, this elegant innovation of randomizing block producers, we feel, is a powerful enhancement.



1. A new list of 21 active witnesses is generated.
2. There are two periods during this time: commit and reveal.
3. During the commit period, each witness candidate generates a random number RN and sends its one-way hash H(RN) to the blockchain.
4. During the reveal period, they send their random numbers (RNs). Each random number is checked in the blockchain using the H(RN) sent during the commit period.
5. When selection time comes, the revealed random numbers of the candidates are used to generate a unique seed (SUM(Reveal(i)), and then the random number generator randomly selects 21 new active producers.

6. Candidates who don’t send a RN during the reveal period cannot be selected.

We have developed a docker-compose configuration that demonstrates this scenario. When a node runs, 63 witnesses are activated on top of it. Active witnesses are changing every 1 minute in the example. Commit and reveal periods succeed each other on each stage. Each witness automatically provides its own commit of a hash followed by a reveal number.

A sample file provides details:

1. Votes for 5 witnesses upon initiation.
2. Continuously prints an account of each witness who produced a block.
3. Prints a number of selected active witnesses after each maintenance overturn (selection).

Layer-1 Prototype

Repositories can be found here:
https://github.com/Revolution-Populi

Implementation

Because this layer-1 is really just a decentralized data facility, transaction speeds can be lightning fast by comparison. Simply put, the mainnet doesn’t get clogged up by a bunch of unnecessary traffic. With this new layer-1, there is just a distributed, decentralized database for different apps to connect into and to share data decentrally, and yet commonly, without betraying or compromising the user’s data.

Each blockchain account can be controlled using a small secret key stored on the user's device - in what we can call a wallet. Keys are used to sign and send transactions to the blockchain. They can also be used to obtain access to content.

Components

Core Blockchain

The structure includes the Core Blockchain based and a cloud storage “starter kit”. The starter kit is for “ease of demo” purposes, and uses a standard mainstream commercial storage just for these purposes. The end system can employ a secure and decentralized storage system (or even multiple systems) such as the InterPlanetary File System (IPFS) or some other similar module or modules. More generally, any app will be able to employ different storages.

RevPopJS

This is the client library for the Core Blockchain, which enables other applications to use the layer-1 blockchain. This idea is commonly referred to as the “DApps” construct in the blockchain space. It’ll be open for any app to use, with open options for tech stacks, pipelines and services.

RevPop Samples

We’ve also included sample files demonstrating different RevPop blockchain scenarios.

How To Run

In short, you need to build and run the Core Blockchain, then install and run sample scripts.

Build and run the Core Blockchain

The easiest way to do this is by using a Docker image. Instructions on how to build the blockchain on your own are available in the repositories.

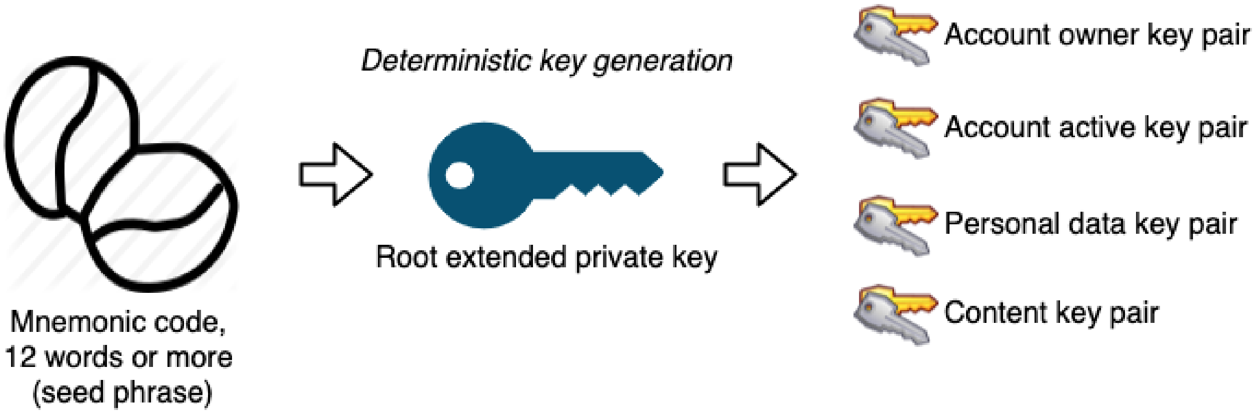
Build and run the Prototype

You can use a pre-configured Docker Compose environment to run all components (in minutes).

System Architecture

Key Generation

Keys are generated from a memorable seed phrase. A user can get access to their account from another device simply by re-entering the seed phrase (while users mainly use their individual mobile devices for their activity, which is personal and secure, and typically in a person's possession, we've also made accommodations for users who wish to use the system on different devices that belong to them, e.g. a tablet).

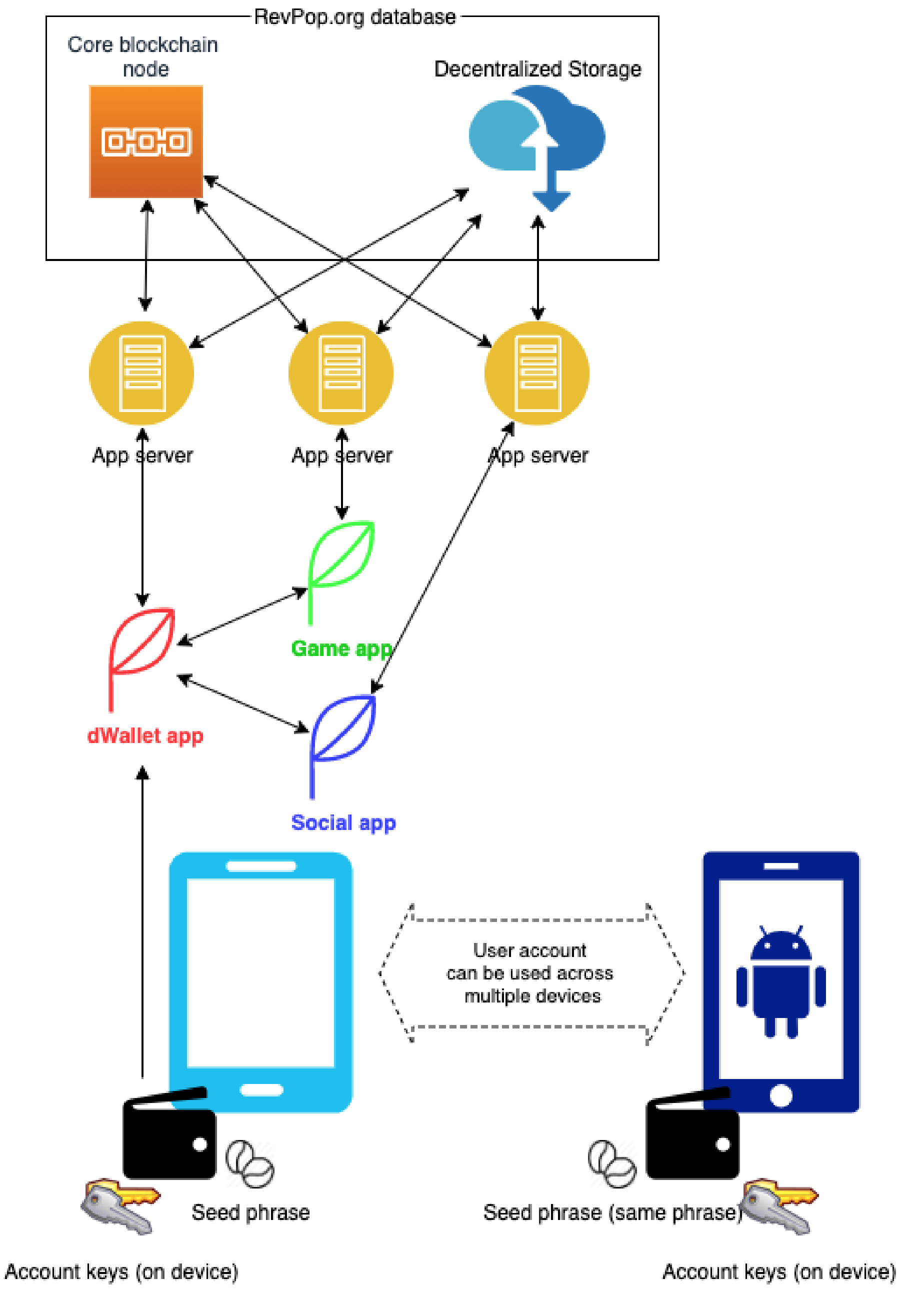


To protect the private key of a user, we've devised an option, but not a requirement, whereby a decentralized, distributed and secure "hot storage" application, based on Shamir secrets between nodes or some other advanced cryptographic mechanism, can be developed and implemented to store keys for other apps to reference. Let's call this a "dWallet" app or just the "dWallet". Other applications can then sign transactions on behalf of a user through such an app, which would have to be done with the consent of a user. These private keys may otherwise just be stored on a user's device directly, and DApps (e.g. social networks, games, etc.) can just access the keys directly.

Ecosystem

Applications would be installed on a user’s device. The dWallet app creates a blockchain wallet for the user. All app data, except the wallet keys, can then be synchronized between devices using cloud.

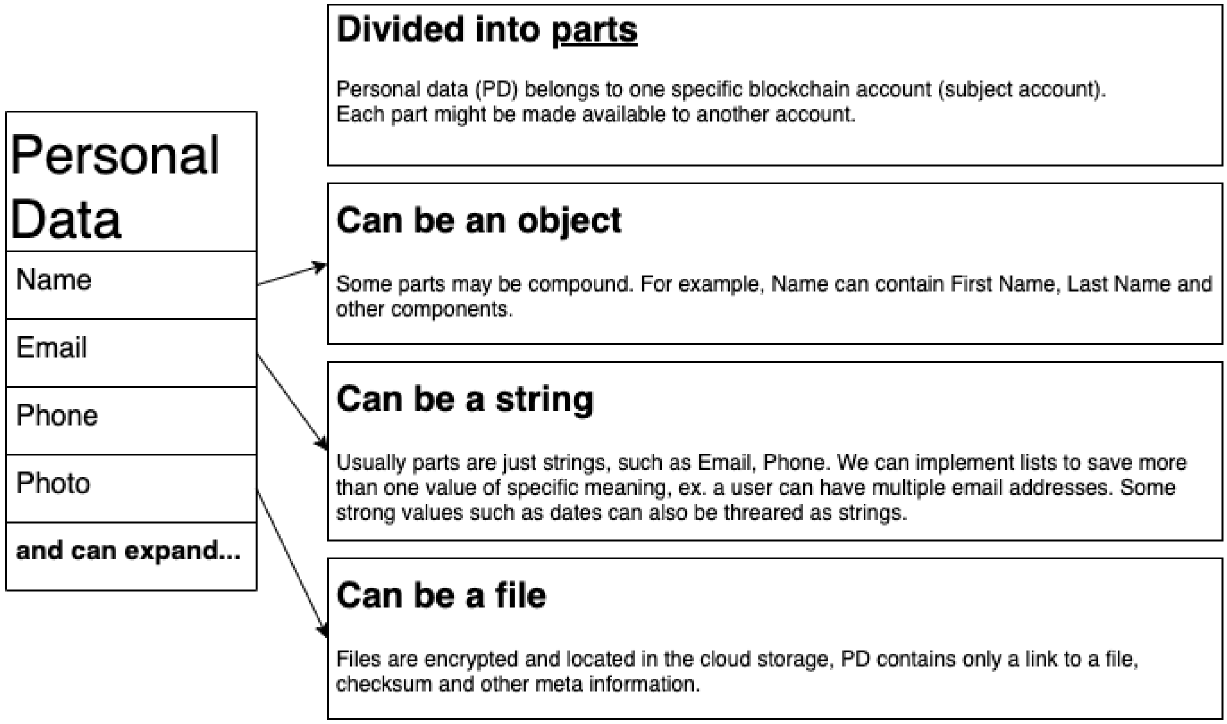
Applications can communicate with the blockchain directly or through an app server. They can read data from the blockchain as well as from the cloud storage (or decentralized storage, as the case may be). Restricted information can only be unlocked from the user’s device, or the application server (but only if explicit permission is granted).



<u>Users</u>

Registering for the First Time

This demonstrates how a user registers in the RevPop ecosystem for the first time using a RevPop-powered app. Here’s how the Personal Data (PD) is structured:



Divided into parts

Personal data (PD) belongs to one specific blockchain account (subject account). Each part might be made available to another account.

Can be an object

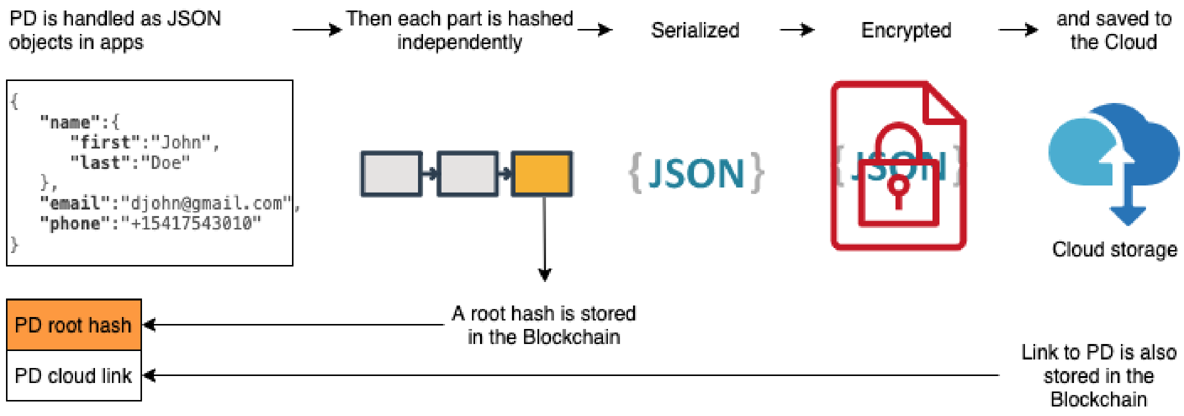
Some parts may be compound. For example, Name can contain First Name, Last Name and other components.

Can be a string

Usually parts are just strings, such as Email, Phone. We can implement lists to save more than one value of specific meaning, ex. a user can have multiple email addresses. Some strong values such as dates can also be threared as strings.

Can be a file

Files are encrypted and located in the cloud storage, PD contains only a link to a file, checksum and other meta information.


PD is handled as JSON objects in apps
Then each part is hashed independently
Serialized
Encrypted
and saved to the Cloud
{JSON}
{JSON}
Cloud storage
A root hash is stored in the Blockchain
Link to PD is also stored in the Blockchain
PD root hash
PD cloud link

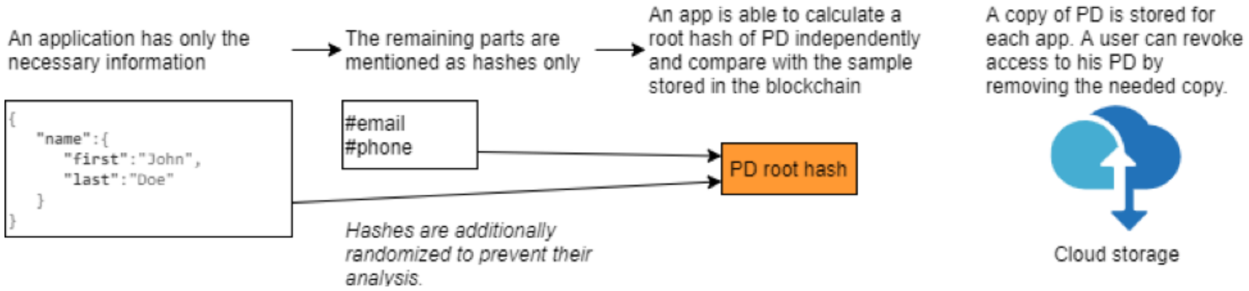
```
{
   "name":{
      "first":"John",
      "last":"Doe"
   },
   "email":"djohn@gmail.com",
   "phone":"+15417543010"
}
```

Sample files (available at https://github.com/Revolution-Populi):

1. case1-onboarding.js - an example of how to create a new blockchain account for a user. Then a user saves his personal data in the blockchain and allows access to the application. Some preparation steps are required before running this sample file. They can be performed by running case0-bootstrap.js.
2. sample-1-account.js - implements the following scenario:
    a. Generate private/public keys
    b. Get balance object and claim balance by registrar account
    c. Upgrade registrar account
    d. Create a new user account with keys generated
3. sample-2-personal-data.js - implements the following scenario:
    a. Save personal data photo to the cloud storage
    b. Create full personal data and sign with the root hash
    c. Save full personal data to the cloud storage
    d. Save full personal data record to the blockchain
    e. Load full personal data + record + photo from the blockchain and the cloud storage
    f. Verify full personal data with the root hash
    g. Create partial personal data and sign with the root hash
    h. Save partial personal data to the cloud storage
    i. Save partial personal data record to the blockchain
    j. Load partial personal data + record + photo from the blockchain and the cloud storage
    k. Verify partial personal data with the root hash

Using Other DApps

The following demonstrates how a user shares PD with another layer-1-powered DApp:


An application has only the necessary information

The remaining parts are mentioned as hashes only

An app is able to calculate a root hash of PD independently and compare with the sample stored in the blockchain

A copy of PD is stored for each app. A user can revoke access to his PD by removing the needed copy.

```
{
  "name":{
    "first":"John",
    "last":"Doe"
  }
}
```

Hashes are additionally randomized to prevent their analysis.

Various parts of existing personal data can be used for signing in to another app; and more information can be added through other apps as well. New apps can verify personal data using a root hash published in the blockchain.

Sample files:

1. case2-registering.js - an example of how to register in another layer-1-powered application using the personal data already stored in the blockchain. Some preparation steps are required before running this sample file. They can be performed by running case0-bootstrap.js and case1-onboarding.js
2. sample-2-personal-data.js - see the previous section for the details

Permitting Other Apps to Publish Content

"Content" is represented in the blockchain by a Content Card.

A Content Card contains:
- Owner account
- Url in the cloud storage
- Publication time
- Type (image, document, etc.)
- Checksum (for verification)
- Encrypted key (private, available to Owner only), used for content decryption
- Description, optional, can be used for indexing

To allow another account to access the content, we create the record in the blockchain.

Content permission information:

- Owner account (who granted a permission)
- Operator account (to whom permission is granted)
- Content card identifier
- Encrypted key (private, available to Operator only), used for content decryption
- Timestamp when a permission is granted

Each content card and permission can be added/updated/removed via a blockchain transaction.

The sample file sample-3-content.js implements the following scenario:

1. Save the encrypted content to the cloud storage
2. Remove the content card
3. Create the content card
4. Remove the permission
5. Create the permission
6. Read the permission
7. Read the content card
8. Load the encrypted content from the cloud storage
9. Update+read the content card

Liking Content

1. Any account can vote for ("like") any content. Adding a vote means sending a transaction to the public blockchain.
2. Information about who voted for what is private by default.
3. A user can share all his votes with any other account. This account can see all user's past and future votes.
4. Any user can see the total number of votes for specific content.

These properties of the system are achieved by a special design where each vote is processed by a randomly selected master node. A master node accumulates different votes, mixes them and publishes a generalized update. This is a powerful and confirmable structure to determine the total number of votes. The next sample checks that these votes (“likes”) go to the right place.

The sample file sample-4-votes.js implements the following scenario:

1. Create content cards
2. Find witness to choose the master node
3. Vote for the content
4. Read and check the vote counter of the content
5. Share voting information with another account
6. Read voting information of another account

Key features of the RevPop Blockchain

1. RevPop's blockchain topology introduces a simple method for storing, securing and serving immutable data, owned and controlled by each user.
2. The system is designed to allow the speed of existing centralized social networks, security through decentralization, full control of user data to the user; and a structure whereby apps can build as they see fit, using tech stacks and services that make sense for them (as an example Chainlink’s decentralized network for data transmission).
3. The blockchain protocol is set up for off-chain integrations both for developer flexibility, but also to limit unneeded throughput to the blockchain. Confirmation speeds can be upheld, less costly systems can run confirmation processes, and the network would still maintain a first rate level of security.
4. DApps would be able to access the layer-1 blockchain through preconfigured APIs that include user validations for data access. Users of these DApps can then have options for different levels of access on a per app basis. Once the user grants access, the data is

then integrated with the layer-1 database, with each person’s data remaining theirs and theirs alone.

5. RevPop's system can offer a new paradigm for the money generated from data. Power over content and data would be vested in the hands of the data generator, where it rightfully belongs; and therefore the ability to monetize the data would too.

Starter Kit Software

Summary

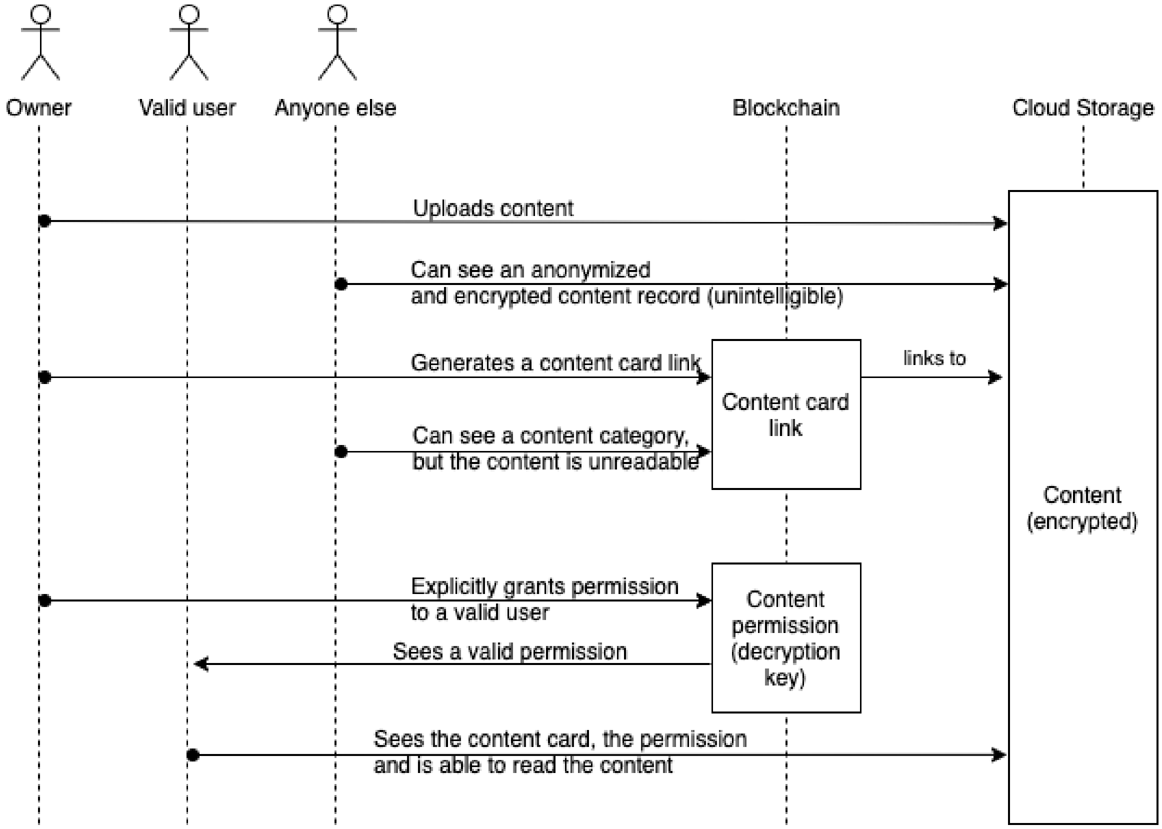
The published prototype is a starter kit, with more to come, beginning with the following:

1. Personal data of a user.
2. Content files.
3. Memorializing and counting user votes.

All this information is stored in the layer-1 database. Heavy content resides in cloud storage, while the link to it and other lightweight content lives directly in the blockchain.

Information stored in the layer-1 database is not available to everyone, since it is encrypted. Only selected individuals and apps can retrieve it by the owner's permission. An owner can do this by sending a special permission object containing a decryption key to the blockchain.

The following diagram shows the process of uploading content and granting access to it:


Owner
Valid user
Anyone else
Blockchain
Cloud Storage
Uploads content
Can see an anonymized and encrypted content record (unintelligible)
Generates a content card link
links to
Content card link
Can see a content category, but the content is unreadable
Content (encrypted)
Explicitly grants permission to a valid user
Content permission (decryption key)
Sees a valid permission
Sees the content card, the permission and is able to read the content

Conclusion

Many different services can be implemented on top of this new layer-1 in a way that is secure, well-functioning, and that foremostly honors data sovereignty.

Disclaimer: This paper is meant to advance discussions and provide some technical guidance, and is not meant to promote any token sales or equity investment by or in Revolution Populi Limited or any of its affiliates or subsidiaries. This paper is in draft form and may change over time.