BrightChain and The Revolution Network:

Bright Block Soup

The Revolution Network / Bright Block Soup / BrightChain

Hi, I'm Jessica Mulein. I spent a number of years essentially in the dark reinventing the blockchain- a little like Leibniz and Newton developing calculus independently. I work at Microsoft on the blockchain now and I've been assembling a number of years of ideas and separate projects into essentially one big project which I'm about to describe.

It has previously been called a few things:

- The Cryptographic Theme Park Protocol
- The Revolution Network
- Bright Block Soup (the underlying block store)
- Five by Five Social (the social aspect)

All of these provide analogies to describe the total picture. After having hashed out several analogies, Bright Block Soup makes the most sense for me to use in explaining what the Revolution Network is. I will use it to try and explain.

I think ultimately what is going to shake out is this:

- BrightChain is the technology (like BlockChain) used behind The Revolution Network
- Five by Five social is likely going to be the front-end/face of the Revolution Network
- People will access blocks from brightchain.net
- People will access the site from fivebyfive.social
- Site info/splash pages will be at brightchain.net/therevolution.network

At its core, the Revolution Network is not an app or any one app. It is a protocol/network. It provides an incentive driven ecosystem that inherently is designed to bring out the best collaborators in the network. Users are incentivized by the algorithms of the network to be philanthropic, collaborative, and provide content and resources to the network. It also provides a unique and, prior to this, proprietary mechanism to provide true anonymity while still maintaining an ability to moderate the network. Aberrant behaviors in the network are disincentivized and penalized.

How is this accomplished? Let's move on to Architecture.

Part 1: The Soup Market

I mentioned Bright Block Soup. Let's start a soup store together.

Some years ago, the Owner-Free Filesystem "OFFSystem" was conceptualized but ultimately failed to gain traction due to lack of participation. Additionally there was not enough data on the overall overhead of such a system. I'm going to explain the Owner Free Filesystem in the context of our block soup.

Why do you care to learn about this? This method of storing not only allows secure data to be held out in the open without security concerns, but it also essentially eliminates copyright concerns for node participants storing data within the network. One thing the world constantly needs more of is storage. Although providing storage to the network isn't required, doing so will be incentivized as we'll cover in a little bit.

Now, imagine you have a bowl of Alphabet Soup. In it are a bunch of jumbled letters that don't spell anything unless you do some work to arrange them. You can then re-arrange them into a new sentence and yet it's still the same soup. So what does that soup really spell out? Does the can you buy on the shelf inherently spell anything? If I dump a can out on the counter, it's not going to make a sentence once- let alone in a repeatable manner.

With that sort of thinking in mind, lets continue.

When we're packaging soup, we like to put it all in cans of the same size. But we do have different size cans like the family size and the standard 8 oz. Let's say we have a can of very special soup that we can dump out and it spells our message exactly. In order to keep the soup looking random, everything that gets put in must be randomized. Therefore we have a random soup can generator whose sole purpose is to have random cans with mixed alphabet messages ready to go for this. Say now we take the special can with our actual message and mix it using a mathematical operation on called an Exclusive OR (XOR). Each letter is affected in order by the added can of random soup and the resultant can looks totally random but in fact has all of the data you put in, though it is now dependent on the other can. This can looks random, but it is still heavily correlated to the original message. We then mix it with more cans in order to dilute the original flavor/message. We need to keep track of exactly which can types we used and they all must be different. Once the message has been mixed with enough random cans, we can safely put it out on the market and nobody will know what it actually says. We throw away the can of special, original soup.

In order to get your original can of soup back, you need to know which flavor of soups are needed and what order to put them in to be able to reconstruct your original message so you go and get a can with that recipe in it.

- Message M
- Random cans/flavors A B C
- Resultant can/flavor Z
- M ^ A ^ B ^ C = Z

now we throw out M and keep A, B, C and Z. These are different soup flavors.

To get M back, A ^ B ^ C ^ Z = M

Assuming your message is bigger than one can, we keep doing this over and over in order until we've covered the entire message.

Anyone is free to ask the store for a can of A, B, C, even Z. But unless you know the recipe for your original can, you can't get it back- especially if there are essentially unlimited flavors. Someone needs to go to the store with a billion flavors and know to ask for your four.

Now we have an open market where anyone can walk in, add soup flavors and request flavors with no questions asked. It's up to everyone to keep their own secret recipe safe.

Getting back to the network here, we can actually keep your recipe in another random can/block and put that into the network as well since it looks random. This makes long recipes (big files) easier to keep since you'll hopefully only need to keep track of a few cans types in order to keep your recipe. These essentially take the form of magnet links something like https://brightchain.net/blocks?recipe=A+B+C+Z

In order to identify the blocks, they will need an identifier. In BrightChain, the SHA256 of the block's full data packet is the id. There is no reason to store multiple identical blocks. *I suspect that there will be a link to a separate option block in order to deal with optional parameters.

For more information on the math, please check the OFFSystem wikipedia entry linked above. It also has links which discuss the overall efficiencies as measured empirically on the network when it was active.

Now, the Revolution Network has a way to store data out in the open, without encryption- although technically speaking this is a form of encryption native storage. At this point we're able to then start a user database.

Unlike BitClout which decided to store data off-chain due to the cost involved, The Revolution Network solves this with the massive distributed filesystem. Users will be encouraged to contribute disk resources and this will help their Valuation

On to Part 2: Authentication

Part 2: Authentication

Most blockchain systems are based around elliptic curve public key cryptography. The Revolution Network is no different.

A SECP256k1 curve like Ethereum will be used.

The network has a handful of different agents that perform different actions and each has its own key pair, the public key of which will be published on the network and stored in the configuration.

For each agent or user, we create a keypair and store it in the bright block store. The user is responsible for keeping the "recipe"- the block IDs needed to get their private key back and decrypt it. The private key will be stored AES encrypted with their password.

User A wants to login:

1. User A presents the server with their recipe for their key
2. The server returns their key blocks
3. The user asks the server for a challenge against their public key, which is known to all actors in the network. A random phrase is encrypted with their public key- readable only with a reconstructed key.
4. Not only does the user need to know the recipe for their key, but they must also know the password, decrypt their key and then decrypt the challenge and return it to the server to complete authentication.

Messages and data can now be sent to or saved for the user using their public key now that we know they can decrypt it. Their password is never stored anywhere. Inputting an incorrect password will either fail to decrypt the private key or the nonsense key will fail to decrypt the challenge.

Let's move on to Central Authority / The Quorum and Identity which will cover Anonymity and Moderation.

Part 3: Identity and Central Authority - The Quorum

While the entire network is decentralized, there is a segmented unit of central authority called The Quorum.

At its core, it uses generation of a computed amount of Forward Error Correction pre-generated, set aside, and broken up into shards so that only once a majority of the desired/precomputed percentage is acquired, the data can be reconstructed.

If you're familiar with the DNS root registry, essentially a handful of different companies and organizations all run root servers. In The Revolution Network, there will be a central, non-profit legal body and effective board of directors responsible for the data integrity of the network.

Almost everything that will pass over the network will be contained in the bright block store. However, The Quorum has a unique role.

Let's say that User Mallory wants to go run amok on the network and post terrible or illegal things. She is entitled to do so. Free speech is what it is. However, certain things like yelling Fire in a crowded place you can't say. Moreover people just don't like some things. We need a mechanism to be able to let people say things anonymously and yet be able to hold them to account if they've ultimately run afoul of the network code of conduct and/or the law.

When Mallory joined the network, she may have registered aliases she's allowed to use (unique). User Mallory wants to post anonymously. She can either use a registered identity with the system that maps back to her original account (think of these like alias accounts), or she can just erase her identity data. Although the post will show the registered identity, no one not in the Quorum will be able to know those aliases. This is essentially part of the user database. The map of user aliases is stored encrypted in the block store and only Quorum members will have the knowledge of the magnet links to identities.

In TCP/IP, we break data down into packets. Any post on the network ultimately has a header a bit like a TCP/IP header. It has a source and a destination. We will use a technique called Forward Error Correction in order to do a little trick. Normally when you post, the section for your user ID must exactly match the user ID/key you authenticated with or The Revolution Network will allow you to enter one of your registered aliases (simple map) or even erase the ID entirely, but not before generating a chunk of error correction data data that gets stuck on the end which contains enough duplicate data to reconstruct errors in the source information. We generate the checksum with the actual entity/key ID of the user doing the post, before manipulating it to an alias ID or empty. The server accepting the message must inspect the message, ensure the identity is registered to the key or be anonymous and that the user is still entitled to use the network in that capacity. The message is accepted and the forward error correction data is split up into equal pieces amongst the Quorum Members.

The Quorum is a sharded authority. Nothing can be accomplished without a majority vote.

The Quorum is effectively a world government. Each shard keeper must then break that shard up according to their bylaws and then safeguard all that data as the independence of the network depends on keeping the governing bodies nonpartisan and able to make informed decisions together. Together. For those in the back: TOGETHER. With majority votes. Digital contracts on shards.

Lets say Mallory's virus or bad content goes in the network a while and is reported or a FISA warrant is issued. At this point, the Quorum must take a vote whether to turn over the identity according to the bylaws, rules of the countries involved, etc. If the answer is Yea, then the Quorum member provides their chunk of the sharded data. Without a majority of Quorum members, the identity can not be reconstructed.

At some point the FEC data on the posts could even be expired and deleted. It might be stored separately so this can happen. Once the "statute of limitations expires" that data could be deleted and the original identity never recovered.

This section has now provided a means to Identify, Authenticate and Moderate users in the network.

Next we get into the Reputation system, and ultimately how your Reputation is your Valuation.

Part 4: Reputation and Valuation

(This section likely needs the most help)

Like most blockchain technologies, your content has a worth and a cost.

The concept of Proof of Work has been used widely over the years. Essentially, you compute something known to take a certain amount of work to do, and the result of that is verifiable with very little work so someone can easily tell you've come up with the right answer.

Everything in the network is done with a proof of work attached. The network has a minimal work requirement for every transaction on the network. You compute a small hash collision for your transaction. The minimal work requirement is given to the best user(s) on the network. An algorithm sets the required number of bits set to zero depending on the user's behavior. Some users who have been known to act poorly may have to do much more work to be able to store and retrieve nodes in the network. This throttles bad actors.

The more your content is liked and/or consumed, the higher your value in the network and the lower your proof of work requirement (no less than the minimum).

Your content takes up storage, this must be paid to the nodes holding your data (everyone in the network might hold copies). When people consume your content, they are inherently giving it a valuation as being worthwhile. Even a post that causes people to get angry might have value and is none the less interesting to the network. Bandwidth costs are paid in coin essentially to the nodes that ultimately serve the blocks to the end user. Different users may run their own nodes or access any particular node.

Every entity in the network has a reputation. Every entity is as much as possible attempted to be the only existing entity in the network corresponding to a real world person or organization. Bob Smith should not be able to have 5 accounts. Bob has one account with 4 or 5 aliases registered. Bob will register with Amazon/Facebook/Google registration and we'll have his phone number and/or email and name. We'll try to correlate and match up real people as much as possible. Exceptions made where real need exists for separate legal entities.

People who contribute resources like storage and bandwidth may have a higher valuation which is a composite of their perceived value as well as the net result of their contributions and costs (reading/consuming) on the network.

- The math of this still needs to be worked out.

Reputation, Part 2

Some time ago, I watched a series on Amazon called Upload. In it, people constantly gave eachother ratings for things. Thanks for the coffee, the ride, just to be nice-whatever. This got me thinking about a universal reputation system.

As User Bob rates things, (which will have a proof of work requirement commensurate with his current reputation), those ratings themselves have a value based on his reputation and affect the rated people and things concordantly.

Pretty much everything can be represented by a URI anymore. A phone number, an address, a person, a transaction. I propose a rating universe in which you can rate pretty much any object, person, transaction, event or thing on the planet and have it develop its own reputation.

URIs known to be associated with a given entity will get linked (TBD) and the collective valuations of their URIs will factor in.

Good people's comments will matter more. If you're a jerk who just is mean to everyone, your ratings will have little impact on the system and you'll work harder to add them.

Valuation

Maths to be decided.

- I had intended the energy price to be measured in Joules and have a real tie to the actual Joule

I was thinking more language level. The concept of a file doesn't really even exist in my world

Just blocks identified by the hash of their content

Some are meta map blocks some are data

Users keep the address to the top block

Meta block

Replication is by durability and duration

As requested at store time

There's going to be an up front discount or otherwise ongoing charges for blocks that don't pay for themselves in user interaction

Tiny

But gas for stored energy bits really

Want to be cognizant of energy costs

Good players keep low proof of work burdens through reputation

Contributors and creators get diamond like reward over time as content is consumed

Essentially your discount for actually using what your stored

Overpaying for durability or duration you don't need won't pay off

Pay servers for transit of blocks in the form of the proof of work.

Use both costs and pays at different rates

But it's all just energy flow tracking

Joules here

Joules there

On to Part 5: Digital Contracts and Secure Virtual Machine

Part 5: Digital Contracts and Secure Virtual Machine

(to be written)

At this time, BrightChain is focusing on the basics of block access and data consistency. In the near future, CIL/CLR based digital contracts will need to be executed on a virtual state machine of some kind. TBD.

5b: Static Indices Computed as a By-Product of Contracts

- Why scour the web. We have everything here, just pre-compute indices as part of the contract. Cryptographically guaranteed indices? Essentially taking advantage of the database filestore.