Trust Models

One of the most valuable properties of many blockchain applications is trustlessness

: the ability of the application to continue operating in an expected way without needing to rely on a specific actor to behave in a specific way even when their interests might change and push them to act in some different unexpected way in the future. Blockchain applications are never fully

trustless, but some applications are much closer to being trustless than others. If we want to make practical moves toward trust minimization, we want to have the ability to compare different degrees of trust.

First, my simple one-sentence definition of trust: trust is the use of any assumptions about the behavior of other people

. If before the pandemic you would walk down the street without making sure to keep two meters' distance from strangers so that they could not suddenly take out a knife and stab you, that's a kind of trust: both trust that people are very rarely completely deranged, and trust that the people managing the legal system continue to provide strong incentives against that kind of behavior. When you run a piece of code written by someone else, you trust that they wrote the code honestly (whether due to their own sense of decency or due to an economic interest in maintaining their reputations), or at least that there exist

enough people checking the code that a bug would be found. Not growing your own food is another kind of trust: trust that enough people will realize that it's in their

interests to grow food so they can sell it to you. You can trust different sizes of groups of people, and there are different kinds of trust.

For the purposes of analyzing blockchain protocols, I tend to break down trust into four dimensions:

- How many people do you need to behave as you expect?

- Out of how many?

- What kinds of motivations are needed for those people to behave? Do they need to be altruistic, or just profit seeking? Do they [need to be uncoordinated](#)?

- How badly will the system fail if the assumptions are violated?

For now, let us focus on the first two. We can draw a graph:

The more green, the better. Let us explore the categories in more detail:

- 1 of 1

: there is exactly one actor, and the system works if (and only if) that one actor does what you expect them to. This is the traditional "centralized" model, and it is what we are trying to do better than.

- N of N

: the "dystopian" world. You rely on a whole bunch of actors, all

of whom need to act as expected for everything to work, with no backups if any of them fail.

- N/2 of N

: this is how blockchains work - they work if the majority of the miners (or PoS validators) are honest. Notice that N/2 of N becomes significantly more valuable the larger the N gets; a blockchain with a few miners/validators dominating the network is much less interesting than a blockchain with its miners/validators widely distributed. That said, we want to improve on even this level of security, hence the [concern around surviving 51% attacks](#).

- 1 of N

: there are many actors, and the system works as long as at least one of them does what you expect them to. Any system based on fraud proofs falls into this category, as do trusted setups though in that case the N is often smaller. Note that you do want the N to be as large as possible!

- Few of N

: there are many actors, and the system works as long as at least some small fixed number of them do what you expect them do. [Data availability checks](#) fall into this category.

- 0 of N

: the systems works as expected without any dependence whatsoever on external actors. Validating a block by checking it yourself falls into this category.

While all buckets other than "0 of N" can be considered "trust", they are very different from each other! Trusting that one particular person (or organization) will work as expected is very different from trusting that some single person anywhere

will do what you expect them to. "1 of N" is arguably much closer to "0 of N" than it is to "N/2 of N" or "1 of 1". A 1-of-N model might perhaps feel like a 1-of-1 model because it feels like you're going through a single actor, but the reality of the two is very

different: in a 1-of-N system, if the actor you're working with at the moment disappears or turns evil, you can just switch to another one, whereas in a 1-of-1 system you're screwed.

Particularly, note that even the correctness of the software you're running typically depends on a "few of N" trust model to ensure that if there's bugs in the code someone will catch them. With that fact in mind, trying really hard to go from 1 of N to 0 of N on some other aspect of an application is often like making a reinforced steel door for your house when the windows are open.

Another important distinction is: how does the system fail if your trust assumption is violated? In blockchains, two most common types of failure are liveness failure

and safety failure

. A liveness failure is an event in which you are temporarily unable to do something you want to do (eg. withdraw coins, get a transaction included in a block, read information from the blockchain). A safety failure is an event in which something actively happens that the system was meant to prevent (eg. an invalid block gets included in a blockchain).

Here are a few examples of trust models of a few blockchain layer 2 protocols. I use "small N

" to refer to the set of participants of the layer 2 system itself, and "big N

" to refer to the participants of the blockchain; the assumption is always that the layer 2 protocol has a smaller community than the blockchain itself. I also limit my use of the word "liveness failure" to cases where coins are stuck for a significant amount of time; no longer being able to use the system but being able to near-instantly withdraw does not count as a liveness failure.

- Channels

(incl state channels, lightning network): 1 of 1 trust for liveness (your counterparty can temporarily freeze your funds, though the harms of this can be mitigated if you split coins between multiple counterparties), N/2 of big-N trust for safety (a blockchain 51% attack can steal your coins)

- Plasma

(assuming centralized operator): 1 of 1 trust for liveness (the operator can temporarily freeze your funds), N/2 of big-N trust for safety (blockchain 51% attack)

- Plasma

(assuming semi-decentralized operator, eg. DPOS): N/2 of small-N trust for liveness, N/2 of big-N trust for safety

- Optimistic rollup

: 1 of 1 or N/2 of small-N trust for liveness (depends on operator type), N/2 of big-N trust for safety

- ZK rollup

: 1 of small-N trust for liveness (if the operator fails to include your transaction, you can withdraw, and if the operator fails to include your withdrawal immediately they cannot produce more batches and you can self-withdraw with the help of any full node of the rollup system); no safety failure risks

- ZK rollup

(with [light-withdrawal enhancement](#)): no liveness failure risks, no safety failure risks

Finally, there is the question of incentives: does the actor you're trusting need to be very altruistic to act as expected, only slightly altruistic, or is being rational enough? Searching for fraud proofs is "by default" slightly altruistic, though just how altruistic it is depends on the complexity of the computation (see [the verifier's dilemma](#)), and there are ways to modify the game to make it rational.

Assisting others with withdrawing from a ZK rollup is rational if we add a way to micro-pay for the service, so there is really

little cause for concern that you won't be able to exit from a rollup with any significant use. Meanwhile, the greater risks of the other systems can be alleviated if we [agree as a community](#) to [not accept 51% attack chains](#) that revert too far in history or censor blocks for too long.

Conclusion: when someone says that a system "depends on trust", ask them in more detail what they mean! Do they mean 1 of 1, or 1 of N, or N/2 of N? Are they demanding these participants be altruistic or just rational? If altruistic, is it a tiny expense or a huge expense? And what if the assumption is violated - do you just need to wait a few hours or days, or do you have assets that are stuck forever? Depending on the answers, your own answer to whether or not you want to use that system might be very different.