# Notes on cryptoeconomics

Oskar Thoren[1,2]

September 10, 2018

[1]Email: ot@oskarthoren.com
[2]Web: http://oskarth.com

# Chapter 1

# Prelude

Notes on cryptoeconomics and related subjects. Started September 9, 2018.

With a focus towards specific problems that I want to solve. These problems should have more precise problem statement. A rough outline will do for now. These are specific problems that are meaningful and relevant and guide reading, even if more general tools are desirable and there is other problems and stuff that is useful.

Additionally, specific papers and free recall notes of these. Then we'll see where a synthesis makes sense.

Also roughly answer things like: what do you know and what would be useful to know?

Previously did a write-up of A Layman's Introduction to Cryptoeconomics.

**Questions**:

1) Incentivized Whisper nodes, including mail server. General problem: full node incentivization and running without a cluster or servers at all.

See things like: Miners; rent proposal; full node incentivization, as well as altruistic models (scale arguments, E.O. Wilson tangent?). Also worth checking re BT and Tor nodes, etc. Any form of p2p.

Litmus test: A system survive without a cluster or subsidised servers.

2) DAO and compensation, funding of public goods and the likes.

Should probable be phrased as question. Finding a good question statement is useful. See Polya for examples of ways of thinking structurally about these problems.

Litmus test: organization/community survive without core contributors/paycheck.

3) More generally, how to balance decentralization/scalability/security? This is essentially Ethereum 2.0 and grokking it at a fundamental level.

# Chapter 2

# Readings

Anything by Vitalik or Szabo is a good start and meaty. Have a bunch in inbox. Meta: what specific ways of reading is useful? Something like the following to keep in mind:

- Who are the actors in this system? - What's the set of rewards and punishments? - What tools are used and introduced? - Is there any empirical basis for this or is it still a theoretical construct? - Lindy effect: Comparable that's 30y/10y or so old? Otherwise maybe BS.

## 2.1 Vitalik - Minimal Slashing Conditions (2017) [redacted]

link

Interest of this is around learning how to think about *slashing conditions*, i.e. punishing bad behavior. Probably lacking some pre-requsites in *byzantine fault tolerance consensus algorithms*.

What does it mean for a block to be economically finalized? **Definition**: a block is *economically finalized* with *cryptoeconomic security margin* $Xifitisguaranteedtobeinthecanonicalchainforever, orifitcostssetofactorsatleast$X [redacted] to remove it from the state.

This is key to Casper design. Validators must submit a *security deposit* to participate, and this can get taken away if the protocol isn't followed. Under what conditions is what is referred to as *slashing conditions*.

Example: PREPARE and COMMIT statements. Key to byzantine fault tolerance consensus two phase consensus design apparently, but not elaborated on right now.

If a validator sends two PREPARE messages within same epoch but with different hashes, say, that should be punished. It's easy for an honest validator not to do this.

When is a block hash *finalized*? When, within a given epoch, you get enough messages of type '[COMMIT, epoch, hash]' that the validator's deposits add up to 2/3 of the *active validator set* deposits.

There are some conditions that slashing conditinos have to meet. One is around *accountable safety*, which says something like: if two hashes gets committed (fork) then at least 1/3 of validators can be penalized. Other is around *plausible liveness*, which is something like: system shouldn't be stuck, there has to be a set of messages that can be sent that 2/3 agree on without validating slashing conditions.

Let's be a bit more formal.

What conditions does a *slashing condition* need to satisfy? Accountable safety and plausible liveness.

What does Vitalik mean by *accountable safety*? If two conflicting hashes gets finalized, it must be (provably) true that at least 1/3 of validators violated some slashing conditions. This is the "economic finality" idea.

What does Vitalik mean by *plausible liveness*? Unless 1/3 of validators violate some slashing condition, there must be a set of messages 2/3 of validators can send that finalize a new hash without violating slashing conditions.

This reminds me of: Schelling points, as well as structured order for Anki (Q/A).

I assume this is directly taken from BFT literature regarding safety and liveness. It'd be useful to understand these in more depth, along with things like sync assumptions (sync/partial/async). It isn't clear to me how this is directly related to any of the specific problems addressed in top, though it is vital for Ethereum 2.0 in general.

There's examples of safety but not liveness, and v.v. Shows failure modes.

These are like little puzzles. Should be able to come up with answers myself.

Quoting algorithm 1: ¿ Algorithm 1: every validator has exactly one opportunity to send a message of the form ["COMMIT", HASH]. If 2/3 of validators send a COMMIT for the same hash, that hash is finalized. Sending two COMMIT messages violates a slashing condition.

Is this safe and live? Safety: if we have two conflicting hashes...

Quoting algorithm 2: ¿ Algorithm 2: every validator has exactly one opportunity to send a message of the form ["COMMIT", HASH, epoch]. If 2/3 of validators send a COMMIT for the same hash with the same epoch number, that hash is finalized. Sending two COMMIT messages with different hashes with the same epoch number violates a slashing condition.

Is this safe and live?

Not enough time to do this properly. Worth re-visiting though.

The pudding is: you can get safety and liveness, but this is actually quite hard. It requires four slashing conditions, and there's a formal proof by Yoichi here: https://yoichihirai.com/minimal.pdf

These are, roughly: COMMIT REQ (sending commit requires 2/3 prepares), PREPARE REQ (don't quite follow, but ancestor epoch related), PREPARE COMMIT CONSISTENCY (if you make commit you saw prepares, so prepares should ref recent epoch), NO DBL PREPARE (can't prepare twice in one epoch).

I'm curious where this split in PREPARE and COMMIT comes from, surely that's a canonical BFT paper somewhere? Same re safety and liveness. What does distsys MIT curriculum say? (can't find anything at first glance).

More human-memorable of safety/liveness: Safety - something bad will never happen; Liveness - system should always make progress.

This type of reasoning/formal proofs is what allows us to make high level statements such as "live under synchrony, safe under async" in PBFT, etc.

There's a lot more to this post, but that's a good start. Worth re-visiting.

**Reading lists** Distributed systems, solid: https://pdos.csail.mit.edu/6.824/schedule.html

Questions: - What's a good primer to byzantine fault tolerance consensus? - What classes of problems do you actually need consensus for? Not clear it is always needed.