Opportunistic consensus

A system that can work when consensus can be obtained, and when it is inaccessible.

The system maintains consistent behavior. The withdrawal of consensus does not break the system according to a certain definition of 'breaking'

A system that can work with multiple sources of consensus.

A consensus is defined as, from the perspective of obtainer, with a given **seed data**, with finite steps, a globally unique state can be obtained, with regard to a time point (physical time, assuming Minkowski spacetime), in material reality. $f(t) = s$

$$\mathbb{C} \text{ from the obtainer} : (\text{Seed}, \text{Procedure}, t) \to C(t) := s_t$$

The notable difference from typical algorithmic paper is that it involves physical time as an inherent, important, part of the description.

This definition involves material reality, which means it's not a construct that only involves abstract ideas.

This is a practical definition, that is to say, we don't care how it is achieved. We don't talk about consensus mechanisms. We talk about effects, the probability that such consensus fails, etc. This is the power of algebraic reasoning.

Blockchains match this definition, while web-of-trust systems don't. Wot heavily depend on parameters that vary by person. Blockchain only requires a genesis block as seed data.

This notion is formulated because the stronger system, always-consensus, is, practically speaking after this much surveying, hard to implement given goals stated below.

$$\mathbb{C}_b : \begin{cases} S = \text{genesis block} \\ P = \text{getting blocks and verifying} \to C(t) \\ t = \text{desired time of interest} \end{cases}$$

$$\mathbb{C}_b \text{ holds for all observers}$$

Now, here is the analysis of web-of-trust

$$\mathbb{C}_w : \begin{cases} S = \text{some people I found trustworthy from random community} \\ P = \text{getting relevant data} \qquad\qquad\qquad\qquad\qquad\qquad \to C(t) \\ t = \text{desired time of interest} \end{cases}$$

$$\mathbb{C}_w \text{ can vary greatly due to choice of } S \text{ resulting in divergence of } C(t)$$

Some people may suggest that an alternative C_1 is, rather *the* correct consensus.

No I am not postulating that C is subjective, as that contradicts the definition. The definition represents an algorithm that reaches a consensus *per se*, regardless of observer.

So in WoT model, some believe its acceptable that there exist a pool of consensus, where each person can reach one, which is believed, to better suit him.

Eventual consistency

An eventual consistent system reaches agreement when a node receives all the messages that have been sent.

The system consists of states, where each s has an *equality relation* defined by the system.

There exists a *partial order*, such that $\forall s_1, s_2 \exists s_3 \geq s_1 \land s_3 \geq s_2$, which is called a *join*

$$\Sigma = (S, =, \geq, \text{join}, \Delta)$$

In a grow-only set, two nodes having divergent sets can exchange messages,
and reach a *join* = $A \cup B$

A grow-only set is isomorphic to $\mathbb{N}$ on eventual consistency Σ

$$U = \text{all possible set elements}$$
$$S \subseteq U$$
$$\geq' := \supseteq$$
$$\text{join} := A \cup B$$
$$\Delta \subseteq U$$

Typical chat protocols, without central server intervention, like Matrix, are not eventually consistent

Chat protocols tend to adopt a directed acyclic graph where nodes are signed messages with receipt of earlier messages.

A join is only possible when a message that merges all branches exist. Alternatively, redefine *equality*, or produce virtual states.

Is it possible to construct a distributed data structure, and reduce finding the join, to be a search problem.

Or structure it such that, after receiving a states, a `max` can be guaranteed.

$$t \text{ for epoch, an interval of physical time as in physical reality}$$
$$e = (t, s)$$
$$\text{For } t', \max_{\forall e(t=t', s)}(s) \Rightarrow \text{the consensus at } t'$$

We can adopt an empirical constraint of max.

Blockchains do not satisfy this requirement, ie, lack of absolute finality. They can only constrain it by threat of slashing.

Absolute finality

$$S_t = \{e \mid (t', s) \land t' = t\}$$
$$\exists \max(S_t)$$

If S_t is finite and unchanging, $\max(S_t)$ is a constant. It can not represent a state.

If S_t is infinite, $\max(S_t)$ does not exist.

We must make $\max(S_t)$ encode a state we want to convey.

Honest signaler assumption

The signaler may release any message from the set, $s' \in S_t$

We model observer as a node with state $S_o \subseteq S_t$

$$S(\text{Current state}) = \max(S_o)$$

Can we make $\max(S_t)$ dependent on S_o. This may encode a useful state, but observers may not converge due to difference in S_o

The trivial case of $\mathbb{N}$ counter

The signaler may release any $n \in \mathbb{N}$

$$\nexists \max(\mathbb{N})$$

The observers take $\max(S_o)$. No shrinkage of S_t can be expected.

Finite state transducer

If the observers can be modelled as FSTs, an output string would be produced after finite reception of messages. The state changes are the messages, in the order of hash-chain order.

$|S_t|$ decreases as $|S_o|$ increases

First message wins

For t_1, a protocol can define the first message from the signaler to be accepted as consensus, because $|S_t| = 1$ and *by honest signaler assumption.*

This is, however, not that useful.

I should revise the assumption. In many cases, there is the the *reversion attack.* A key gets stolen and it's used to reverse a consensus.

This can be easily done, with a *state oracle*

1. An authoritative server
2. State proofs from some blockchain

re-Formalization of the hashed DAG

Nodes are messages, which are often state deltas, and *ultimately* represent states.

Links based on cryptographic hashes are proofs of *temporal/causal precedence*

DAG is a set of states with a partial order. Note that this comes with something beyond math. The block linked by another block can only be created, first.

Branching attack / Reversion attack

The central problem in the state synchronization turned out to be branching.

S_o is defined to be the internal belief of a node about some state.

Theoretically, after t_1 passed, State before t_1 shouldn't change anymore, by definition.

The node may ignore further messages that change the past state, but new nodes can be "deceived".

The only solution to branching attack is "coercion", or "will" that complies with the protocol

For coercion, blockchain offers most coercion, and it can be very efficient.

A blockchain with 1M dollars at stake can handle infinite amount of state.

For will, you can use a "trust" based mechanism to pick *complying* sources.

Consider a blog that has one signer and is stored as a DAG. The blogger is benign to himself, ie, he doesn't attack his own blog.

One day he lost his key. The attack erases the entire blog by signing an alternative branch with empty content.

The two branches are inherently non-comparable states. Old nodes can ignore the new message, but new nodes get deceived.

The simple solution is to, finalize every blog change to a blockchain, ie. include the blog state in the blockchain state root.

Now, to attack a small blog the attacker has to go against the entire blockchain.

Isn't this cost-efficient?

Further, we can see that blockchains with more $ in stake and more decnetralization are more trustworthy. We can publish state hashes to multiple chains, and when we check for results, they are weighed according to the trustworthiness calculated by the-above-justifications, as a weighed voting.

TODO: Review blockchain state proof candidates

Recalling from meory, there are a few

- ZK proofs that simply prove entire chain of state transitions up from genesis, eg. Mina.
- Multi-signatures from the quorum, eg. Ethereum.
  - I couldn't find the post. Anyway it was about having Eth stakers sign a state root, such as signature is enough because the production of a false signature is deterred by slashing.

Notice that blockchain state proofs do not need any additional communication. They can be verified within the contract.

> The consensus layer light client conforms to the beacon chain light client specification, and makes use of the beacon chain's sync committees (introduced ahead of the Merge in the Altair hard fork). The sync committee is a randomly selected subset of 512 validators that serve for 27-hour periods.
>
> When a validator is on a sync committee, they **sign** every beacon chain block header that they see. If more than two-thirds of the committee signs a given block header, it is highly likely that that block is in the canonical beacon chain. If Helios knows the makeup of the current sync committee, it can confidently track the head of the chain by asking an untrusted RPC for the most recent sync committee signature.
>
> — https://a16zcrypto.com/posts/article/building-helios-ethereum-light-client/

So here is the TODO, I'll be making a lib for verifying and publishing such checkpoints.

TODO: How do contracts use them

- Contracts can use them by themselves, indenpdently.
  - I should write some lib for this.
- Aggregate them to save gas fee on chain?

A contract can be seen as a state machine with possibly branching state changes.

$$\begin{tikzcd}
 & s_1 \arrow[dl] \arrow[dr] & \\
s_2 & & s_3 \arrow[d] \\
 & & s_4
\end{tikzcd}$$

The basic way to use blockchain for *state pinning*, is to publish the state root onto a chain immediately after posting, to prove its timestamp.

This is very rudimentary. Does not require any on-chain logic, but only uses the chain as a witness of time.

Much more can be done, ofc.

In comaprison, L2 solutions batch states changes and post the proofs of the transitions on chain, so that they can be used on chain. This is unlike L2. We use the chain as an external service.

Decoupling the chain has benefits, we can use any chains as the *timestamp witness*, any chains as microtransaction handler.

Web of trust

We characterise blockchains as innately trustworthy agents that follow the protocol, with the nuance that, they vary in degrees of reliability.

- If, say, a blockchain has stakeholders that are in secret collusion. They can collectively defy the protocol. The goal of the field has therefore been optimizing towards a state apparatus where all the members know nothing but to follow the protocol.
  - Culminating in Nullchinchilla's https://melproject.org/en/

Hereby, we define blockchain as an institution, wherein members are reduced to cogs and rules prevail, which as a whole is an *agent* with some will, some expected behavior to outsiders.

The other *orthogonal* perspective is to not make innately *aligned agents* with regard to our goals.

The story starts from that when you use a usual web service, you trust a single corporate operator, which they call a single point of failure.

Going up the metacognitive hierarchy we know it's only conceptualized as a single operator, but in reality it's under the influence of internal corporate employees and various dynamics.

Alignment

Analysis has shown that all of that is ultimately directed towards *alignment*. We want a blockchain to follow a protocol; we want trusted CA roots in our browser to act “honestly”.

Either by coercion or voluntary alginment from the other side, it's going to be done.

The society constantly does a web-of-trust thing, the credit ratings, the connections underlying business world, the bureaucracy of tangible state.

Consider, in the new social platform, you have a list of subscribed news channels. You maintain the list to kick out “malicious sources” that publish ADs, etc.

As a form of decision making

- You may adopt information from other trusted agents, to efficiently remove more malicious sources
  - which forms the graph structure of web-of-trust

As a state apparatus

- You coerce people into behaving to your liking, ie, *alignment* by the threat of unfollowing.
- You maximize coercion by forming coalition with other people
  - which necessitates the graph structure
- Like any state apparatus, web-of-trust is self-prophetic.

Hereby I pointed out the metacognitive underpinnigs of the seemingly innocent idea of web-of-trust.

Avenues of improvement

The part of coercion can be furthered by forming a larger governance structure where a ban happens decisively. Sure, this institution can go corrupt, or undesirable as in my views.

There is a lot that can be done to engineer a better *institution* in this realm of computation.

- Anonymous voting that is decentralized
- Anonymous voting with more complex voting mechanisms

Law of necessary solution

If a good, decentralized solution is not present, and a solution is demanded, people will opt-for whatever is available.

- People go to corporate-run platforms because socialization is a necessary demand.
- Matrix users run centralized ban-lists to keep out malicious actors

I disdain the centralized ban-lists due the above reasons. In any case, this section serves to warn you against ignoring any unsolved-demand. They can go in the ways you hate.

Law of necessary existence

If you don’t make an institution that removes CSAM, your protocol will face existential threat.

If you don’t make an ideal, non-biased institution that removes CSAM, people will invent one themselves and it will probably be biased, corrupt, and centralized.

The problem of web-of-trust can therefore be divided into two goals, if you want to make an algorithm for them.

1. The algorithm and system to evaluate most trustworthy agents
2. The algorithm and system to *punish* misbehaving agents

This may sound a bit weird to some but (2) is a necessary consequence of (1) as a matter of fact.

On implementing Web of trust

I’d rather call it *Egoistic Alignment.* As illustrated above, this is *orthogonal* to the cosntruct of blockchain, or consensus systems.

We consider a list of agents, and assign a probability of them behaving according to some *criterion*.

$$Q = \{N = (\rho, K)\}$$
$$K \text{ denotes the totality of knowledge we have about } N$$
$$\rho = f(K) \text{ we derive the probability of the node well behaving from } K$$

We don't need to constraint ourselves to some specific models of estimation. Why not make use of everything possible to estimate better metrics.

Necessarily, the user agent (the node which the user runs, acts on behalf of the user and represents the user on the network), has beliefs (about what is spam, etc.)

I model it with beliefs, rather than decisions because this notion makes transitive trust easier to model. And it connects with probability theory directly.

$$B_{\text{elief}}(P_{\text{roposition}}) \in [0, 1]$$
$$\text{we use probability theory directly, which is equivalent to fuzzy logic}$$
$$Q = \{N = (\rho : P \to P_N, K, B)\}$$
$$\text{B is the probability the node believes about P}$$
$$\text{we calculate a probability about a node's trustworthiness about the specific belief}$$
$$\rho(B) \text{ therefore means } N \text{ being right about } P$$
$$B \text{ is what } N \text{ asserts the probability about it}$$
$$B = \frac{\sum_{N \in Q} \rho(P) \times B_N}{|Q|}$$
$$\text{this is the weighted average approach, maybe there are better}$$
$$\text{recusrively, we can estimate the proposition, a node being well-behaving with the function}$$

There are 2 metrics to quantify the performance

- The differnentiating power of the model. Beliefs should have large differences, but not overestimating.
- The amount of nodes that can be included in the model.

$$\rho(P) = \frac{\sum_{N \in Q} \rho(\rho(P)) \times B_N(\rho(P))}{|Q|}$$

Recursive functions are seen, but the logic model requires definite values, so it's a constraint satisfication problem.

Anyway, this is a general framework based on philosphical foundation that works with arbitrary functions.

This is a system of constraints that are developed from logic and probability theory. We can reasonably show that a valid estimator must satisfy the constraints, but a specific algorithm is not given; such is the power of algebra.

Provable Quoting

Sometimes I want to quote messages from a person. If it's done with a screenshot, It can not prove the existence of messages.

Trivially, a proof can be generated with ZKP

Commited messages

You can send a message that is revealed in the future. In the interim you can not change it.

https://en.wikipedia.org/wiki/Commitment_scheme

Timecapsule

You can send safe timecapsules into future wihtout relying on an central message keeper.

https://www.drand.love

Case analysis: A service offering anonymous exits

Exits can be abused. When abuse happens they lose a huge portion of value because they got blacklisted from web services.

It's not just about payment. There is an intrinsic, and huge, component of trust.

We want it to be

- Sufficiently anonymous, since it's a system for, eg. paid Tor exits.
  - This part can be done with ZK.
- Sufficiently enforcing. We want to minimize abuse.

I can see there is a fundamental difference in assumption of identity in an *anonymous trust system*

Note, If you want to survery the literature, the search words would be

- "modelling trust in distributed systems"
- "reputation systems"
- "bayesian web of trust", etc.

The analysis presented above has cyclic probability which is probably computation intractable. I expect existing literature to be a simplification of my analysis.

They tend to assume a continual, pseudo-anonymous identity, where each of them has a track history of behavior. Pseudo-anonymity links online activity together, which is a compromise of anonymity.

If, we can transform a picture of total-anonymity trust data into pseudo-anonymity trust data, we can input it into well-studied trust models.

Formalization. Totally-Anonymous Trust System

The whole process preserves an anonymity of $n \in N$ anonymity set, if we consider the simple, anonymity set conception of anonymity level.

Identity Initialization $\rightarrow$ Activity $\rightarrow$ (end)

The process is cheap, and a user may make a new identity for every post he makes.

The trust system may publish some data for the Initialization as long as it doesn't weaken the anonymity. Equivalently, we can say the system maintains a state.

Typically, the trust system publishes a ZK-proof according to its protocol needs. Again, the information revealed preserves the anonymity level.

The Initialization must provide enough information/proof for other agents to calculate his trust scores.

There are indeed papers on ZK and anonymous reputation systems. Engineering-wise, we should prefer writing ZK-algorithms in a ZK VM. Prefer existing ZK frameworks over equivalent but manually written cryptosystems. Prefer composing existing gadgets and using frameworks.

The state-of-art ZK systems are more performant than stated in most papers.

https://github.com/privacy-scaling-explorations/zk-eigentrust/blob/master/docs/4_algorithm.md

We can rewrite this in latest iter of ZKVM.

https://nlp.stanford.edu/pubs/eigentrust.pdf

Attempt 1

We may consider each pseudo-anonymous identity to be a collection of unlinkable, anoymous identities.

For each operation directed at a pseudo-nym, we replace it with a vector of anonyms.

$$\text{Proving ownership of a pseudo-nym} \rightarrow \text{Proving ownership of a vector of anonyms}$$

I will try using this idea to port existing Trust models. EigenTrust produces the same global values for every peer. The simple idea is to use ZK-membership proofs to prove membership in buckets of high-ranking anonyms.

In other words, it's equivalent to proving that "you made a post that has among top 10K-100K in upvotes, and you made a project that is among top 1K to 10K by stars, and ..." which means, proving a pseudo-anonym is treated as a conjunction of anonyms.

If we can obtain a same vector (for every node) of trust scores in ℕ, we prove k memberships in k buckets that have high trust scores. This reference vector simplifies the proving.

> It should be emphasized that the pre-trusted peers are essential to this algorithm, as they guarantee convergence and break up ma- licious collectives. Therefore, the choice of pre-trusted peers is important. In particular, it is important that no pre-trusted peer be a member of a malicious collective. This would compromise the quality of the algorithm. To avoid this, the system may choose a very few number of pre-trusted peers (for example, the designers of the network). A thorough investigation of different methods of choosing pre-trusted peers is an interesting research area, but it is outside of the scope of this paper.

I have not found a satisfying decentralized, anonymous, reputation system in the literature yet.

Limit the number of anonymous identities

We'd want the number of anonymous identities to be finite, and maybe rate-limited.

This is doable with ZK. In general producing a unique hash for every use of an anonym, and proving the lack of such hashes from a merkle tree.

TODO.

Attempt 2, Formalization

We characterise a pseudo-anonyous identity, aka. *pseudonym*, as a continuous presence on Internet.

$$\text{Pnym} = [s_1, s_2, s_3, ...]$$
$$\text{link}([s_n]) : \text{public}$$
$$\text{Token} = (\text{pubkey}, \text{signature})$$

$[s_n]$ are states published in public, where the link between them is also public. The token is used to prove identity.

It's a form of currency where any malicious behavior depreciates the "token" drastically and continual compliance is heavily rewarded.

For an *anonym*, the link is only known to the user.

$$\text{Anym} = [s_1, s_2, s_3, ...]$$
$$\text{link}([s_n]) : \text{private}$$
$$\text{Token} = (\text{trapdoor}(\text{secretkey}), \text{proof})$$
$$\text{each token is unlinkable, indistinguishable from other tokens}$$

Fundamentally we can be sure that, all models of reputation are mathematical manipulation of the available information. Everything not based on *information* are guesses.

I don't think there is a need to fixate over a specific model, because anything that constructs such a currency should work.

In a typical reputation model, the $[s_1]$ is converted into $n \in \mathbb{N}$, a score, where s_n is a trade and the peer's rating about it. I consider this a loss of information.

Consider

$$\text{Anym} = [s_1, s_2, ..., b_1, b_2, ...]$$
$$s_n \text{ represents good things he has done}$$
$$b_n \text{ represents bad things he has done}$$
$$\text{both are backed by state that exists in public}$$

To prove a good reputation, you construct a ZK-proof to show you have done many good things, and a exclusion proof of bad things.

Transitive trust without pseudonym

$$\text{Pnym\_rating} = f(\text{pnym}) \to n$$

This is the typical way a peer expresses trust about other peers, but it doesn't work without continual identities.

In an anonymous system, the trust graph is built over s_n. The edges connecting identities (originally) are distributed over the states s_n.

This method just turns a psuedonym into many small pseudonyms. Is this good enough?

There is a lot of metadata leakage although it's splitted.

Ontology of Trust

The untrusted thing must be logically connected to something that is trusted in the first place. The logical connection is called **inference**. There is no way an untrusted thing can be trusted without a logical connection. ZKPs are "zero-trust" due to a complex construction of mathematics, which is in itself, rigorous logic.

The various reputation models emphasize 'transitive trust' because there is no other way to derive trust.

Whim: Is it possible to prove the logical inference, logical connection in zero knowledge

$$\text{prove that } s \text{ is a descendent of } a, b, c, ..., \text{in a trust graph, in zero-knowledge}$$

There is a lot of information leakage already?

In other words, it's trying to prove that "you should trust me because you trust X, Y, Z, and I am somehow related to X, Y, Z through some relationships that confer trust"

This can not be turned into a cryptocurrency problem either, because the value of trust coming from each node, is subjective and changing.

Does an anonym expressing endorsement for other s_n (anonyms) compromise anonymity? If so, can we avoid this.

Can we construct a graph that is isomorphic, and make it public instead?

Anonymous graph isomorphism

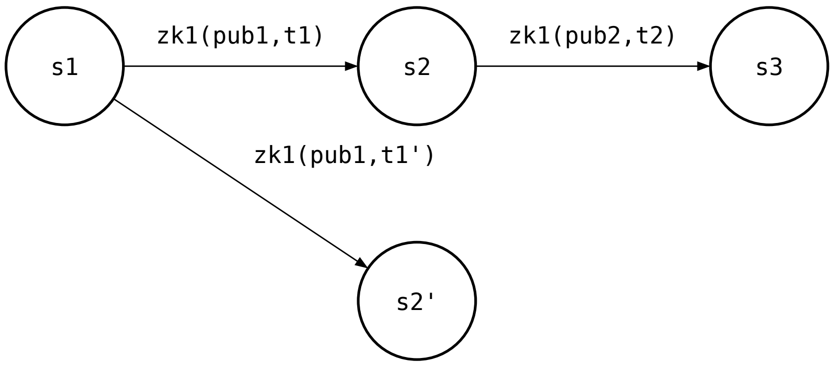
We define a trust graph, where nodes are assigned trust values, directed edges represent conferring of trust.

$$\text{CompTrust}(N) \to N'$$
$$\text{where } N \text{ has nodes with 0 trust and } N' \text{ has a trust assignment for every node}$$

Provenance of trust is done by proving the ownership of nodes. Think of each node as a publickey, provenance is showing a signature. In classical web-of-trust, each user owns one keypair, and proves trustworthiness by using publickey-cryptography.

We consider graph isomorphism that maps graph A to B, where one node may be mapped to multiple nodes, so that the trust relationships are obfuscated. We want it hard to reconstruct the orginial graph.

Whim: a general scheme based on ZK and state machine



It's possible to construct a graph with ZK, precisely

$$G = (N = \text{States}, E = \{(\text{ZK-Circuit}, \text{Input})\})$$

or alternatively

$$G = (N = \text{Input}, E = \text{States})$$
$$\text{for now I dont talk about this one}$$

etc.

Each s_n represents a set of beliefs of trust. All t_n are secret.

Make the user trust the initial state s_1 of the state machine, and every possible state following the execution.

The circuit in use should uphold the rule of ‘transitive trust’. Thereby, each s_n represents an accurate set of beliefs about trust.

Would this work? Seems like a good general construction.

When a user in the middle of the whole trust graph wants to express trust about some other users (as identified by posts), he produces ZK proofs of state changes, *from other nodes* to the desired terminal state that reflects the change. Thereby hiding his expression of trust.

The said circuit should ofc require the user to prove its reputation, from that very state, so the user could produce the output state which includes his desired *transfer of trust*.

The scheme must provide a method to merge states, so one set of trust beliefs can be produced.

This system tends towards a minization of states kept on network, as they can be merged and eliminated.

Paper