Farcaster is a prominent alternative to centralized social media. Similarly to Semaphore, Farcaster puts identities on an external chain (Ethereum) that is synced by Farcaster nodes, or "hubs." Like an alias, a Farcaster identity consists of a unique number, a pubkey, and a human readable name. Unlike Semaphore, Farcaster uses only a P2P network without a consensus mechanism to propagate messages. Farcaster has the stated goal of being a "sufficiently decentralized social network," meaning that users have "the ability to claim a unique username, post messages under that name, and read messages from any valid name."

Farcaster also describes itself as "an open protocol…Just like email." This is a problem for Farcaster because as explained previously, email's decentralization was not durable and failed with usage. What killed email as a decentralized network is spam. Farcaster is just as vulnerable to spam as email and will be doomed to the same centralized fate if it sees meaningful usage. The Farcaster design does not seriously consider what is necessary to build a decentralized layer for network effects and instead only decentralizes identities. The tradeoffs in this design are not worth it. The primary factor keeping disgruntled users on Twitter is NOT that it is too difficult to find users on other platforms; the issue is that Twitter controls the network effects. In fact, this makes the centralization pressure on Farcaster far greater because email, which is one to one communication, does not have network effects.

The designers of Farcaster implicitly understand that a unified network is necessary: if users cannot reliably connect, they will coalesse onto a single platform. Farcaster hubs are designed to sync their state between different machines, but syncing state is not viable without a consensus mechanism. Spam prevention is especially at odds with a shared state in the Farcaster design.

One solution would be to not bother: have all nodes maintain their own state and have users simply send messages to the nodes they want to serve their messages. This is the approach taken by nostr. As nostr says in its readme, "[nostr] does not rely on P2P techniques, therefore it works." However, this obviously means that network effects are separated to each node and are entirely under the control of the node operator. This of course doesn't solve the problem of decentralizing network effects and Farcaster does not take this approach.

Even before considering spam, syncing separate hubs without a consensus mechanism is a difficult problem. Mastodon users have issues interacting between instances even though Mastodon instances are all trusted nodes and content is associated with a particular instance. Both these points make instance interaction much easier and yet there are still issues, meaning integrating farcaster hubs will be even less reliable. To ensure that running Farcaster hubs is manageable, excessive spam must be prevented. This would require that hubs limit the number of users and limit the frequency with which users can post. The creation of new Farcaster identities is limited only by the gas cost on Ethereum. If Farcaster hubs wish to prevent spam, the only limit on the creation of new identities is the incidental scaling limitations of Ethereum rather than explicitly designing limits. This is not a viable solution. Even granting a limited set of aliases, Farcaster does not have the necessary systems to limit the broadcasting of individual identities. Each hub must implement its own rate limiting rules. Since Farcaster does not have a consensus mechanism, each hub is incentivized to be more permissive. If users are being rate limited by Alice's hub but Bob's hub does not enforce a rate limit, users are more likely to switch to Bob's hub, causing centralization. If Bob can pay for the infrastructure to run a super-hub it may be in his best interests to try to increase throughput so that other hubs cannot compete. This process of centralization into the hands of a few large service providers is exactly what happened to email. Bob's ability to capture the network effects just like twitter makes this an even bigger problem.

Even if hubs agree on a rate limiting policy, they are still likely to fall out of consensus. Farcaster messages do not have reliable timestamps: they are user submitted with no way to verify. Instead, a hub must measure against its own clock when deciding to reject a message. Desynchronization between clocks and network latency means that nodes can easily fall out of sync by receiving messages at different local times or receiving them out of order. For instance, if timestamps that are 30 seconds late are rejected, a broadcast with a forged

timestamp can be strategically released so that half of the network receives it before the cutoff and half the network receives it after the cutoff. As a result, spam prevention results in a highly unreliable network state. Additionally, unreliable timestamps mean that a Farcaster hub cannot distinguish between a message with a forged timestamp and a message that it was late to receive. Forged timestamps further complicate spam prevention and make it impossible for a hub to reliably know if its state is synced. If there is sufficient noise on the network, hubs may never converge on a shared state! To mitigate this, hubs round times down to the nearest 10th second, but this can similarly be manipulated by forging timestamps. Desynchronization between clocks and network latency means that preventing forged timestamps will also lead to an unreliable network state. Furthermore, offline nodes would have no way of knowing which messages had genuine timestamps. The only solution without a consensus mechanism would be to implement trusted nodes from which to sync. Semaphore solves this issue by timestamping everything via the epoch vote and using karma to rate limit. However, karma only works because nodes are in consensus about which messages are accepted and when, necessitating reliable timestamps.

The keys for Farcaster identities are stored on Ethereum and are used to authorize signers. If a signer is revoked (such as because the key is lost or compromised), all messages with the old key must be deleted, otherwise messages with forged timestamps will still be accepted. Similarly, if the ethereum address changes by a user executing an account recovery, there is no way for hubs to know if a message was signed with the old key before or after the update because there are no reliable timestamps. When validating a Farcaster broadcast, hubs do not have a reliable way of knowing the relevant Ethereum state at the time the broadcast was originally sent. This is an issue because knowing the Ethereum state is necessary to know which public key should be used to verify the broadcast. Semaphore solves this issue by including the external chain headers in each block, removing any ambiguity.

In theory, the benefit of Farcaster's design over Semaphore's is that it is simpler. However, the issues outlined above mean that in practice Farcaster greatly complicates its network by eschewing a consensus mechanism. Consider email again: SMTP is a very simple protocol but running SMTP in an adversarial environment is enormously complicated because problems must be handled externally to the protocol. Despite having none of the benefits of a blockchain, Farcaster suffers from the biggest downside of a blockchain: validation cost. Every message in Farcaster is cryptographically signed. Avoiding a blockchain does not actually improve scaling constraints because each signature must still be validated! Because Farcaster does not have any mechanism to limit total throughput, a hub does not know what the worst case computation/bandwidth load would be. Because Semaphore uses a blockchain, nodes know exactly what the worst case computation/bandwidth load is. Furthermore, because Semaphore's design packs broadcasts into discrete blocks, signatures can be easily and efficiently aggregated to a constant size, reducing storage and bandwidth usage.

Semaphore's blockchain also means that users can trustlessly verify the chain without needing to process everything. A user can check the proof of engagement by simply validating a single aggregated signature per block. A user can check the chain tip by participating in a single round of the epoch vote. Although there are reasonable centralization concerns if a blockchain's capacity is very large, these problems are far worse in the Farcaster model. Farcaster hubs can easily omit content from users and censor broadcasts without users knowing. Because the Semaphore blockchain acts as a reliable source of truth, users can easily detect malfeasance from any node.

Although Semaphore is better suited to the use case of social media, this does not mean that Farcaster does not have an important role to play in the future of online social interaction. Farcaster is best suited to smaller scale, non-global communication that does not require synchronization between hubs. A "web3 Discord!" Because all communication on Semaphore is global, a Farcaster-style network would give aliases the ability to DM each other/form group chats. Although this is a different use case than global social media, it is still important. The two networks can even share a set of identities!