So, how does Semaphore actually accomplish any of its goals while solving the issues of other systems? How are the number of identities limited without causing centralized control? How can spam be prevented while still maintaining a unified network? How can a network both be censorship resistant while allowing for moderation? The full explanation can be found in the Semaphore whitepaper, but this writeup should serve as a gentle introduction.

Semaphore is a blockchain without a cryptocurrency. Instead, Semaphore uses “crypto-identities” or “aliases.” Each alias has a unique integer, a pubkey, and a human readable “nym.” The set of aliases is maintained on a separate type 2 network, such as Ethereum. However, aliases are much more than simple NFT’s (non-functional tokens), nodes that run the semaphore network sync the state of the alias smart contract and only accept broadcasts/connections signed by pubkeys belonging to valid aliases. This means that the state of a smart contract on Ethereum can control the set of valid users on the Semaphore network. The rules for the total number of aliases able to be minted can follow a predetermined function, be controlled by an alias vote, etc. The details are beyond the scope of this writeup.

The lifecycle of a broadcast starts when it is signed by a user. The message includes the user’s alias, signature, the message itself, and a timestamp. Time on Semaphore exists as discrete epochs of 6 seconds, so the timestamp says which epoch the broadcast occurs in. Next, the broadcast is relayed to Semaphore nodes, who will relay the message to their peers. This process is called the “initial relay.” A broadcast will be accepted as long as its timestamp is no more than one epoch early or late, meaning a broadcast timestamped from epoch 100, will be accepted in epoch 99, 100, or 101. Each node on the network maintains a network time, which is kept in sync with its peers using a BFT process. Once epoch 102 begins, the initial relay for epoch 100 will end and the “epoch vote” for epoch 100 will begin. At the end of the initial relay, it is possible that nodes will have seen different sets of broadcasts. The epoch vote is a BFT process that will allow nodes to reach consensus on which broadcasts were seen by a majority of nodes during the initial relay. Once the epoch vote has terminated, a block containing each accepted broadcast is constructed by each node. To reach consensus, nodes will iteratively sample each other in a process similar to the Avalanche consensus mechanism. If a supermajority of nodes report that a broadcast should be included in a block, a node’s confidence will increase, otherwise it will decrease. Even if nodes initially disagree about which broadcasts have been seen, the confidences of each node will converge. A broadcast is accepted if its confidence is positive at the end of the epoch vote. At the end of the epoch vote process, with very high probability all broadcasts with a correct timestamp will be included in the block and all nodes will have produced identical blocks.

To run a node that participates in any BFT process (clock & epoch vote), it must have an alias. Additionally, aliases that do not wish to run a node are incentivized to delegate to an alias that does run a node. To manipulate either process, an attacker must control ⅓ of aliases. Attacks are disincentivized because acquiring that many aliases would be expensive and executing an attack requires identifying oneself via the aliases, allowing for social slashing. Additionally, because Semaphore does not control assets like a typical, type 2 blockchain, there is far less profit motive to attack it. Because Semaphore does not have a cryptocurrency, think of nodes participating in the epoch vote not as miners or stakers, but app operators ensuring that their users' posts are written to the network. Unlike PoW or PoS, which are designed to be costly for syble resistance, the epoch vote gets syble resistance for free because participating requires an alias. Therefore because social apps need to run a backend anyways, participating in the epoch vote has a marginal cost of ~0.

The epoch vote on its own is insufficient, however, because it is a subjective process: offline nodes cannot verify the output like they can with PoW or PoS. The missing piece is “proof of engagement.” Each block has a “chain commitment,” which is the hash of previous blocks. The chain commitment is analogous to the previous block hash in a typical blockchain. The chain commitment is also what broadcasts use as a timestamp. Chain commitments mean that if there is ever a blockchain fork, broadcasts are only valid on a single fork. When an

alias makes a broadcast with a particular chain commitment, it is said to have committed to the fork that the chain commitment corresponds to. If there is a blockchain fork that has reached a sufficient depth, nodes will consider the chain that has more commitments after the fork to be valid, meaning it has the higher proof of engagement. Proof of engagement has a few interesting benefits. First, the security of the chain derives from the users of the chain, rather than the block producers. This means that a majority of nodes cannot cause a deep reorg if users do not consent. Second, the chain cannot reorg without deleting all messages after the reorg. This means that even a totally centralized version of the chain (which we are working on because it is easier to release early) has a pretty strong notion of finality because the centralized chain sequencer cannot selectively delete previous broadcasts. Third, proof of engagement can be easily checked by offline nodes. Because Semaphore uses BLS signatures, the proof of engagement for each block can be checked in O(1) time by validating a single signature.

OK, but how expensive would it be to run a node? Actually, the better question to ask is how much more expensive is running a Semaphore node compared to running an alternative system, such as Farcaster? The costs are essentially the same! In both systems, every message must be downloaded and every signature must be validated. This represents nearly all of the costs of node operation. Semaphore has a small overhead for tracking karma and a tiny overhead for running consensus, but this is trivial compared to the unavoidable costs common to any possible system. Furthermore, Semaphore's design allows users to trustlessly verify the chain without downloading everything, which is not possible in a non-blockchain system.

As stated before, one of the purposes of Semaphore's blockchain architecture is to enable spam prevention, which is achieved through Semaphore's karma system. The karma system is a dynamic rate limit that also enables popular aliases to have greater access during times of congestion. The goal of the karma system is to fairly allocate blockspace without the use of fees. A good design will ensure that the total throughput never exceeds the capacity of the network but never throttles users so much that blocks are not full even though users want to post. There are a number of parameters to the karma system that can be tweaked for different outcomes. The design must also ensure that a small clique of aliases cannot manipulate the system to exclude other users from the chain or infinitely increase their own access. The karma system implements a rate limit with an exponential backoff. If the on chain volume exceeds a target capacity, the rate limit increases. An alias increases its karma score, which can be thought of as non-transferable points, by receiving interactions from other aliases. Karma can be burned to exceed the rate limit. The amount of karma that must be burned is set by the protocol, depending on how much the target capacity is exceeded. The details are beyond the scope of this writeup. Finally, the network should be able to operate at a sufficient scale that rate limits usually go unnoticed. Only rarely should the network be congested enough that rate limits+karma impact the user experience, such as if a real-world event leads to large amounts of online discussion. Although there is some complexity to tuning the parameters of the karma system, once the system is established, node operators do not need to worry about which messages are spam and which are OK. All complexity is handled by the protocol.

Another purpose of Semaphore's blockchain architecture is to enable moderation. Moderation is important on social media, especially when users are able to post to a decentralized, censorship resistant system. The Semaphore philosophy is that users should have maximum control over the interactions with their own content and that different users should have the freedom to implement different rules according to their own values. Each alias is able to set a blocklist. If A blocks B, then B is unable to interact with any broadcasts from A unless the block is revoked. Additionally, a broadcast that is not a reply to another broadcast, can define a "paradigm," or set of rules that apply to it and all descendent broadcasts (replies, replies to replies, etc). This means that users can moderate the discussions that take place under their broadcasts. Paradigms are simple scripts that allow a broadcast to be valid if the script returns True. Some functionality includes: blocking aliases in a set, only allowing aliases in a set, blocking aliases that are in m/n sets, not allowing interaction before a time, not allowing interactions after a time, etc. Paradigms enable precise control over how interactions can

occur. We think it is likely that different organizations would maintain sets of aliases that they consider to be spam, misinformation, hateful, etc. Users have the option of using the sets maintained by others in their own paradigms, making moderation significantly easier. Furthermore, paradigms are saved in the blockchain state, meaning that once one person has defined a paradigm, others are free to use it. This even allows for subreddit-like constructions or other moderated communities to be programmatically defined in the blockchain. For instance, a paradigm can define a set of aliases that are banned from the moderated community and allow only a set of moderator aliases to update the set. The paradigm can also define the rules for adding or removing a moderator alias, such as requiring ⅔ of current moderators to agree. Paradigms give users powerful moderation tools, and make the results of their efforts composable, so all users can easily define the best rules for their own unique experience or use a moderation policy “off the shelf.” Moderation of social media often appears to be endlessly contentious, but paradigms offer a solution by making open source, collaborative, and composable moderation possible unlike anything else. If moderation rules are not enforced by the network, then apps are free to ignore the wishes of the users who post the messages. Although all apps are free to implement whatever moderation/curation policies they like, the app does not have the option to ignore moderation policies set by any user, even if the user only broadcasts from a separate app. Lastly, this system makes spam far more expensive than a type 2 network such as Lens. In a type 2 network, all users must pay the same fee, meaning there is an unavoidable tradeoff between expensive posts for honest users, and a network flooded with cheap spam. In Semaphore, honest users only need to acquire an alias once but spammers must continually acquire new aliases as old ones get added to blocklists.

Semaphore's design enables a blockchain that operates without fees or a cryptocurrency, which is the only design fit for a robust decentralized social network. Furthermore, paradigms enable community moderation that is far more powerful than any other system and enable programmatically defined online communities. To see comparisons between Semaphore and alternatives, take a look at the followup writeups!