Semaphore is a decentralized social layer for the internet that enables interoperable social apps. It is designed to be censorship resistant while still enabling users to moderate their interactions. It is designed to be spam resistant while maintaining a unified network of users/content. Although these goals may initially appear to be at odds with each other, a novel blockchain design that works via “crypto-identities" instead of cryptocurrency makes it possible. This means Semaphore’s blockchain can autonomously validate social interactions without requiring fees or gas.

Before understanding the design of Semaphore, we must look at how decentralized networks can go wrong: email. As Bitcoin OG Jameson Lopp explains in his article, The Death of Decentralized Email, “[SMTP] is no longer usable,” having been hacked to death by “many minor engineering decisions” over the years. After its inception in the 1980’s, email was not an adversarial network, so network administrators would altruistically relay messages to help the fledgling protocol. Once millions of people began to use email in the 90’s and advertisers sought to reach them, spam became a major problem. The SMTP protocol was built with the primary goal of reliably delivering messages. Spam necessitated rules outside of the protocol that purposefully discarded certain messages. This included admins closing their relays and closing connections with other open relays, resulting in the use of whitelists and blacklists. However spam was still a problem, so admins implemented filters to prevent certain messages from reaching users’ inboxes. These measures were partially effective but resulted in non-spam messages getting blocked. This necessitated special communication channels between admins, meaning that reliable email delivery relied upon human decisions. The resulting system is far more complex and expensive to operate than was originally envisioned. Although on the surface email was still using SMTP, the “real” protocol was not the simple protocol from the 80’s, but a complex web of company policies, bayesian filters, business relationships, and decades of other hacky solutions. As a result, all but the biggest email service providers, such as Microsoft and Google are priced out. It is now essentially impossible to run one’s own email server. You can run the software, but your emails will not be reliably sent or received. These problems are only more difficult to solve in the social media context: it is far easier to filter one to one communication to an individual’s inbox than it is to filter communications to a global network of users. So, how can these issues be avoided? How could email have avoided these issues?

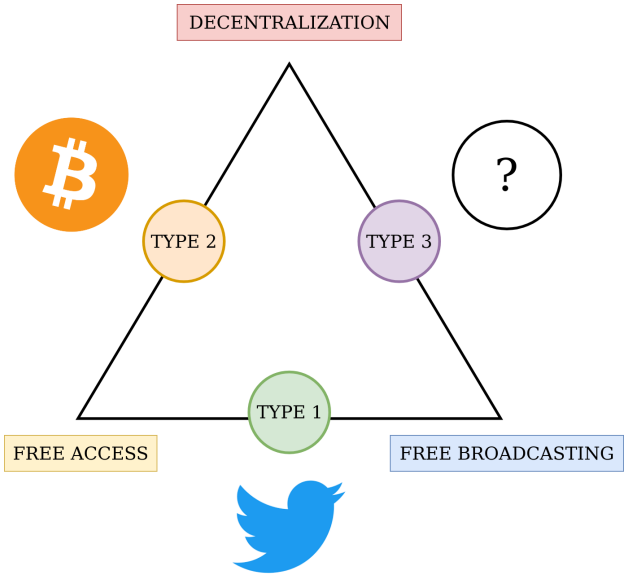
The issue of email spam was mitigated by embracing centralized solutions. Centralized service providers have the capacity to handle a deluge of messages (at great cost) and filter for users. But there could have been alternate solutions. Hashcash, invented by Adam Back, was one such solution, but it failed to see adoption in email. Spam was only possible because emails were free to send, so hashcash added a small cost to sending an individual email by requiring the hash of the email to have a certain number of leading zeros. Although hashcash was not adopted by email, it is used in Bitcoin mining! One other solution would have been to limit the total quantity of email accounts. A limited set of identities would enable effective rate limiting of users, preventing spam, though implementing this without causing centralization would likely not have been feasible at the time. Let's formalize this idea a bit. Option 1 is to limit spam by using centralized service providers, sacrificing decentralization. Option 2 is to make sending messages expensive, sacrificing free broadcasting. Option 3 is to make the total number of users limited, sacrificing free access.

Communication networks operating at scale have three desirable properties, but only two are achievable at a time due to bandwidth, computation, and storage constraints. The three properties correspond to the previous three options for preventing spam:

- **Decentralization**: The ability for an individual user to inexpensively receive and validate network activity without relying on a central party.
- **Free Broadcasting**: The ability for an individual user to broadcast a message to the members of a network at trivial or zero cost.
- **Free Access**: The ability for an individual user to be recognized as a participating member of the network and be guaranteed access to the functions of the network at trivial or zero cost.

We call this the “Network Rights Trilemma” and it results in three classes of networks:

- **Type 1**: These networks sacrifice decentralization. They include email, social media, banking, etc. Making a Twitter account is free and tweeting is free, but Twitter is not decentralized and cannot be made decentralized because an individual cannot run Twitter servers/spam prevention algorithms (even Twitter can barely do these things).
- **Type 2**: These networks sacrifice free broadcasting. They include Bitcoin, Ethereum, and other open blockchains. Bitcoin is decentralized and making a wallet is free, but having a transaction included in a block requires a fee to be paid to the miners. Removing the block size limit would remove the fee, but would sacrifice decentralization by making nodes expensive to run
- **Type 3**: These networks sacrifice free access. Semaphore is the first network of this type, which is how it can succeed in areas where existing type 2 networks fail. How to limit the quantity of users without causing centralization will be addressed soon, don't worry!



SMTP was always destined to fail because it tried to achieve all three properties simultaneously. Because of the incentives of running a business, the only way to protect decentralization is to explicitly make the right tradeoffs in protocol design. The “S” in SMPT stands for simple. To keep the protocol simple, it did not worry about spam, only reliable delivery. But spam is real and the simple protocol resulted in enormous systemic complexity and cost outside of the protocol. The Semaphore protocol is somewhat more complicated than alternatives (a bit more complicated than Bitcoin, much less complicated than Ethereum), but this complexity is encapsulated and makes actually running the protocol (which is what matters) at least an order of magnitude simpler than the alternatives. Of all “decentralized” social media alternatives, only Semaphore’s design considers the trilemma and its impact on social interaction. This writeup is not focused on alternative networks, but some TLDR’s are:

- **Mastodon** is like email today, with instance operators having the same powers as email operators, causing centralization. However, because social media is far more complicated than email, there is not a reliable unified network between instances.
- **Farcaster** is like email in the 80’s. To the extent that it functions today, it is because it is not operating at scale. Its design attempts to create a unified network, but without a consensus mechanism, this cannot be done reliably, especially when accounting for spam. This will result in a fragmented network at best, but is likely to also become centralized like Mastodon or email.
- **Nostr** is also like email in the 80’s. Unlike Farcaster, it explicitly does not have the goal of a shared network between relays. Nostr accepts a fragmented network, but relays are still under the same centralization pressure as email, Mastodon, and Farcaster

- **Lens** is different from the other because it is a type 2 network running on polygon. However a type 2 network is not a good fit for social media because users have a low willingness to pay for social media broadcasts. Social media posts are fundamentally different types of goods than financial transactions, which type 2 networks are ideal for. Additionally, Polygon sacrifices decentralization for lower fees. But lower fees also means it is easier to spam. Because everyone pays the same fee, an honest message is always just as expensive as a spam message, but a type 3 design can ensure that non-spam is cheaper than spam!

To learn how Semaphore's type 3 network design solves these problems and enables novel functionality, take a look at the next writeup!