Programmable Cryptography and Ethereum

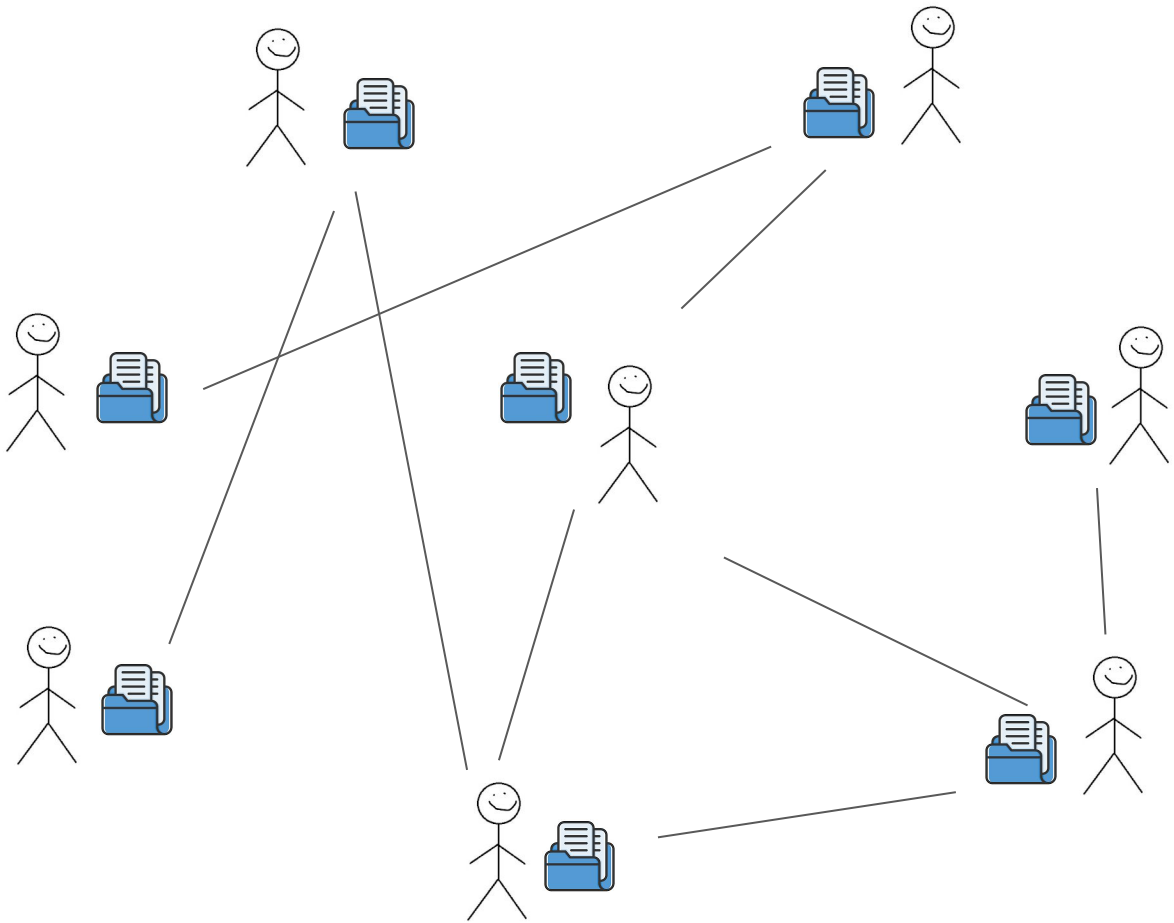
gubsheep - Devcon SEA

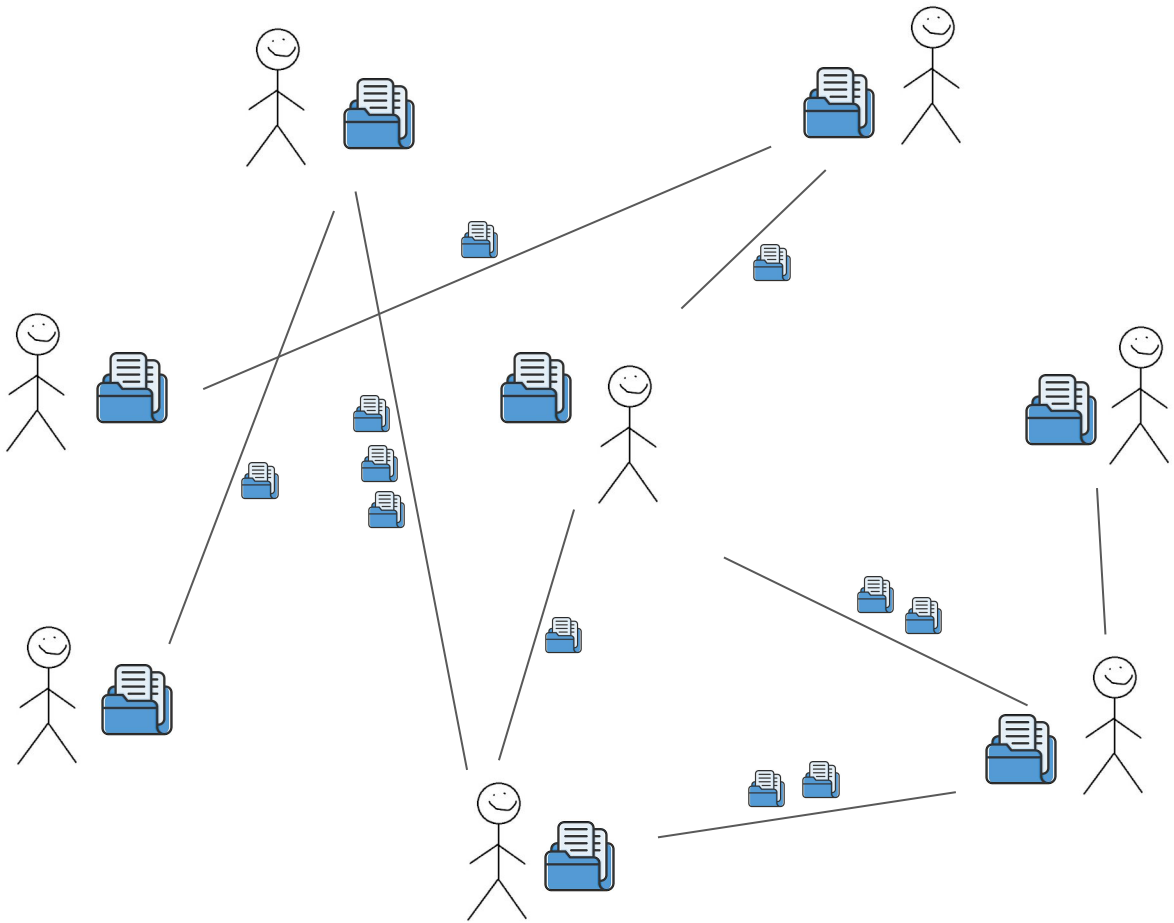
0xPARC

INTRO

Gubsheep

How do **we** compute **together**?

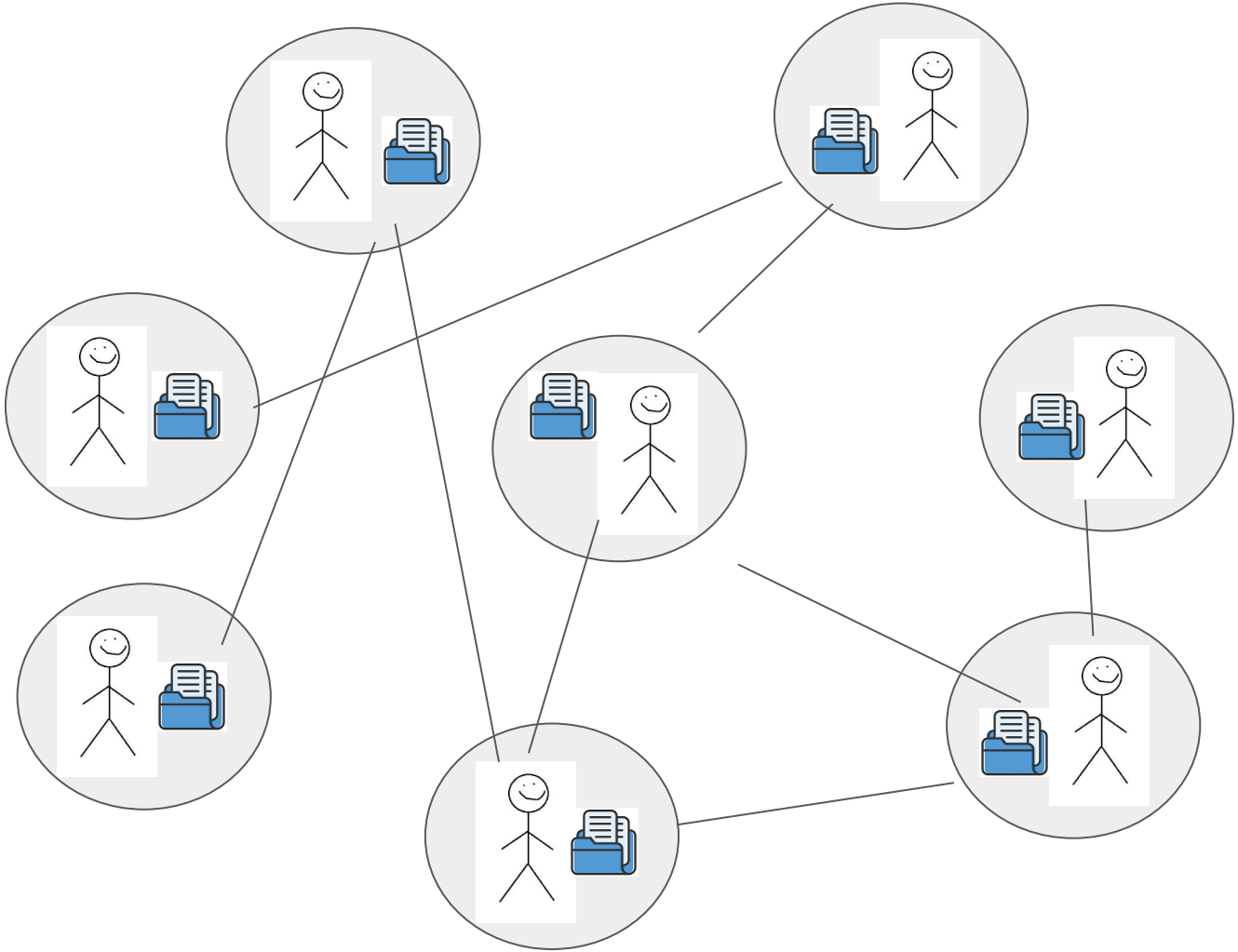


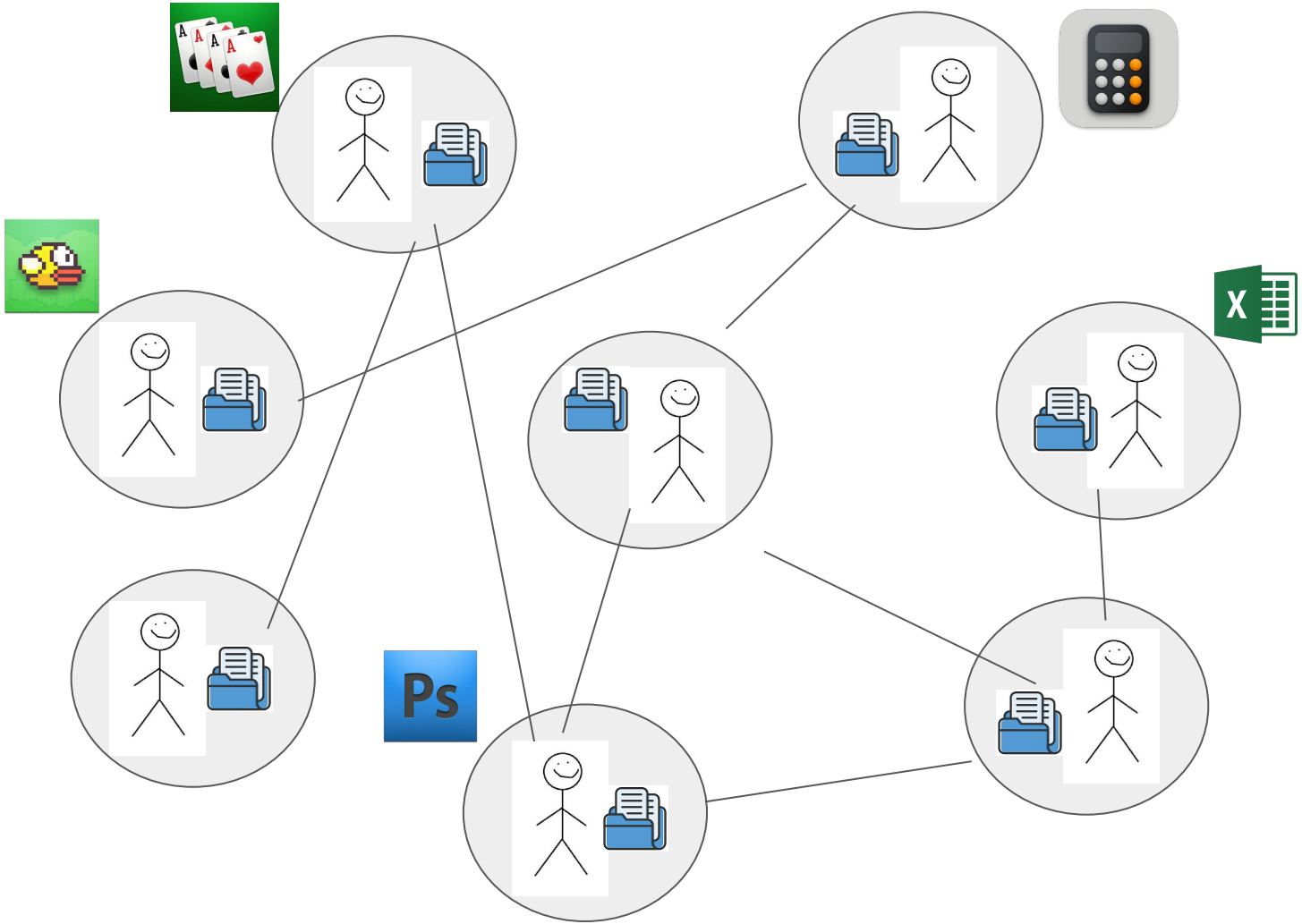


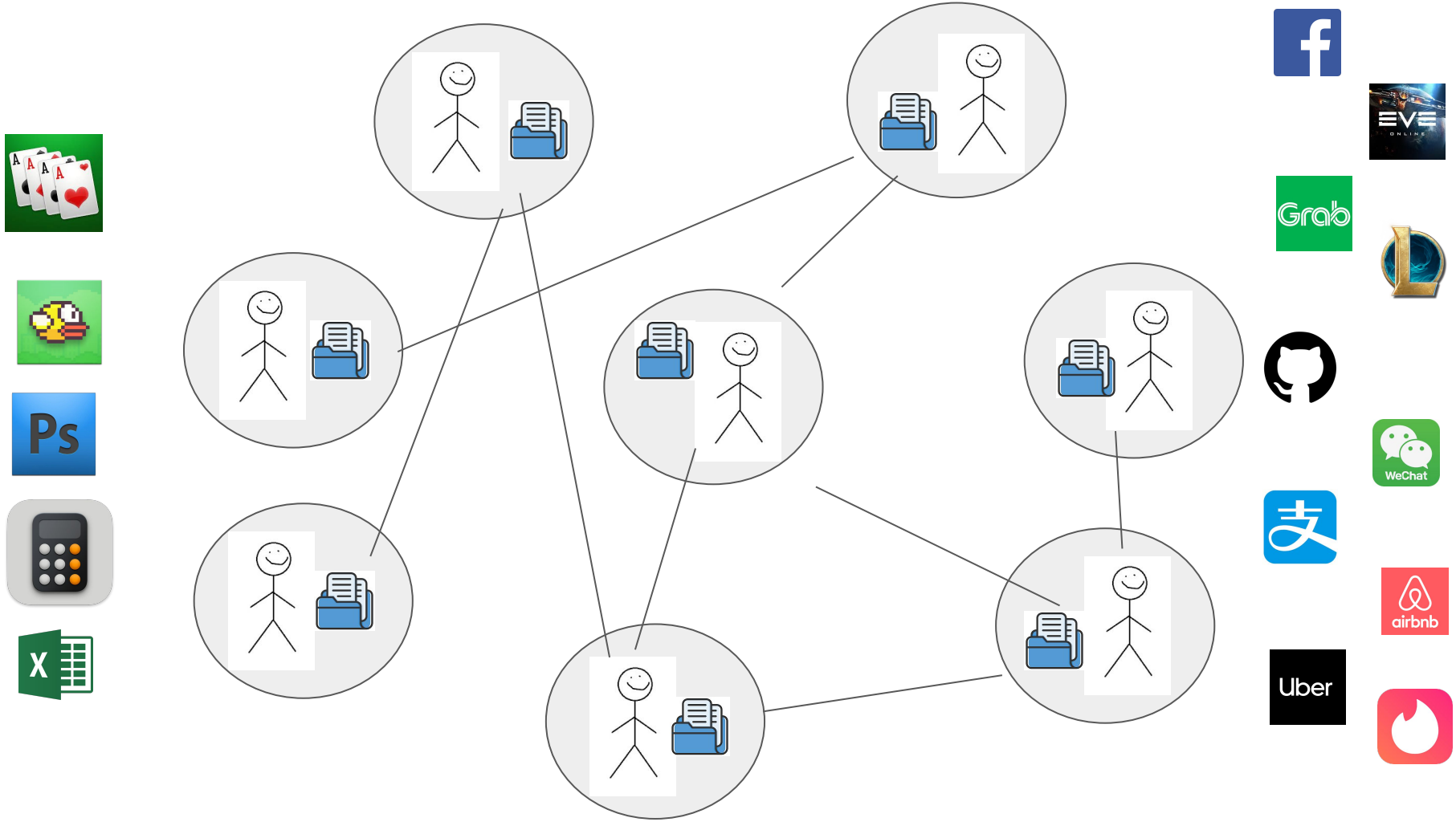
Anonymous February 23, 2009 at 8:46 AM

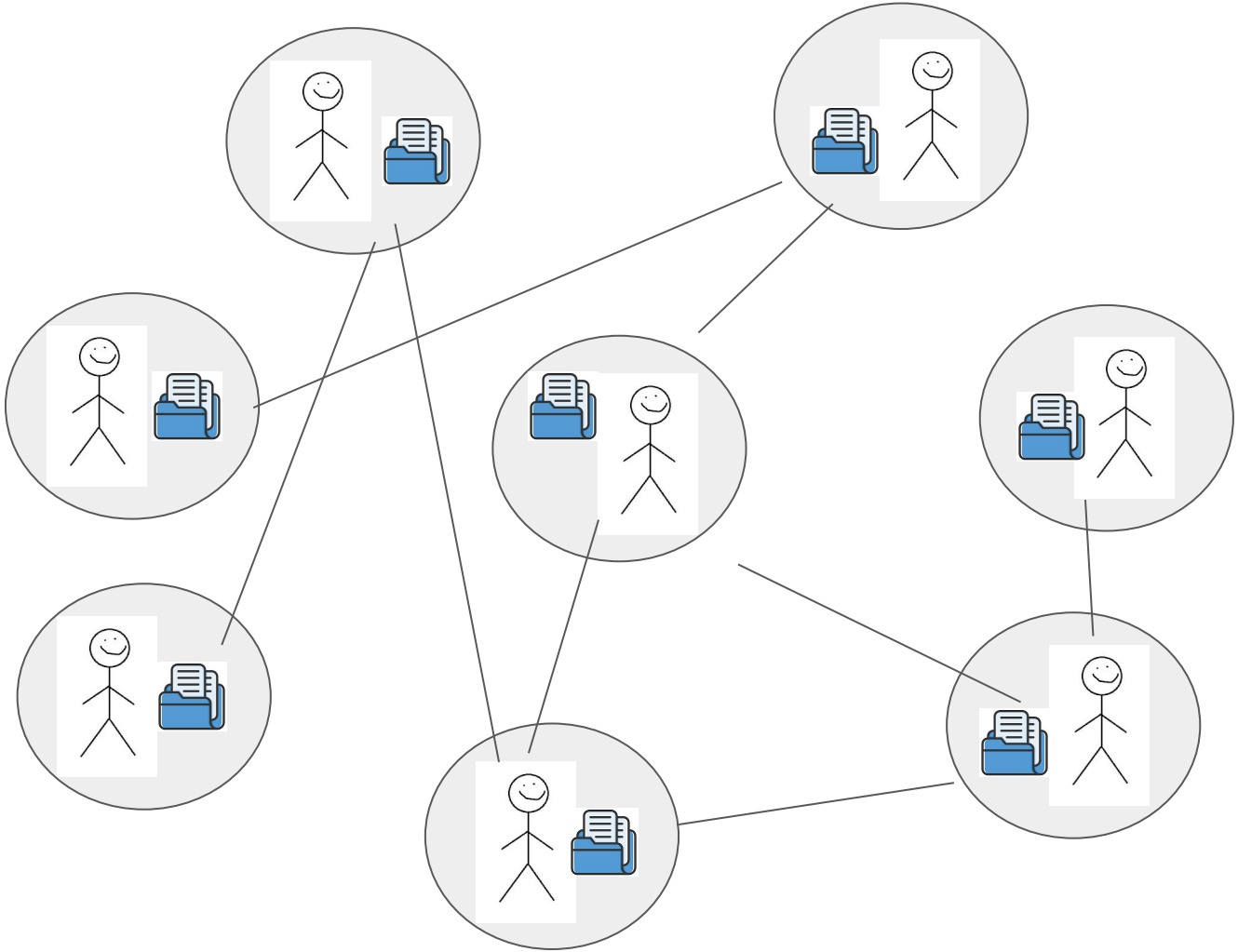
this is hilarious. reminds me of when i first saw the web. my reaction was: ftp + mouse. so what?
-Remzi

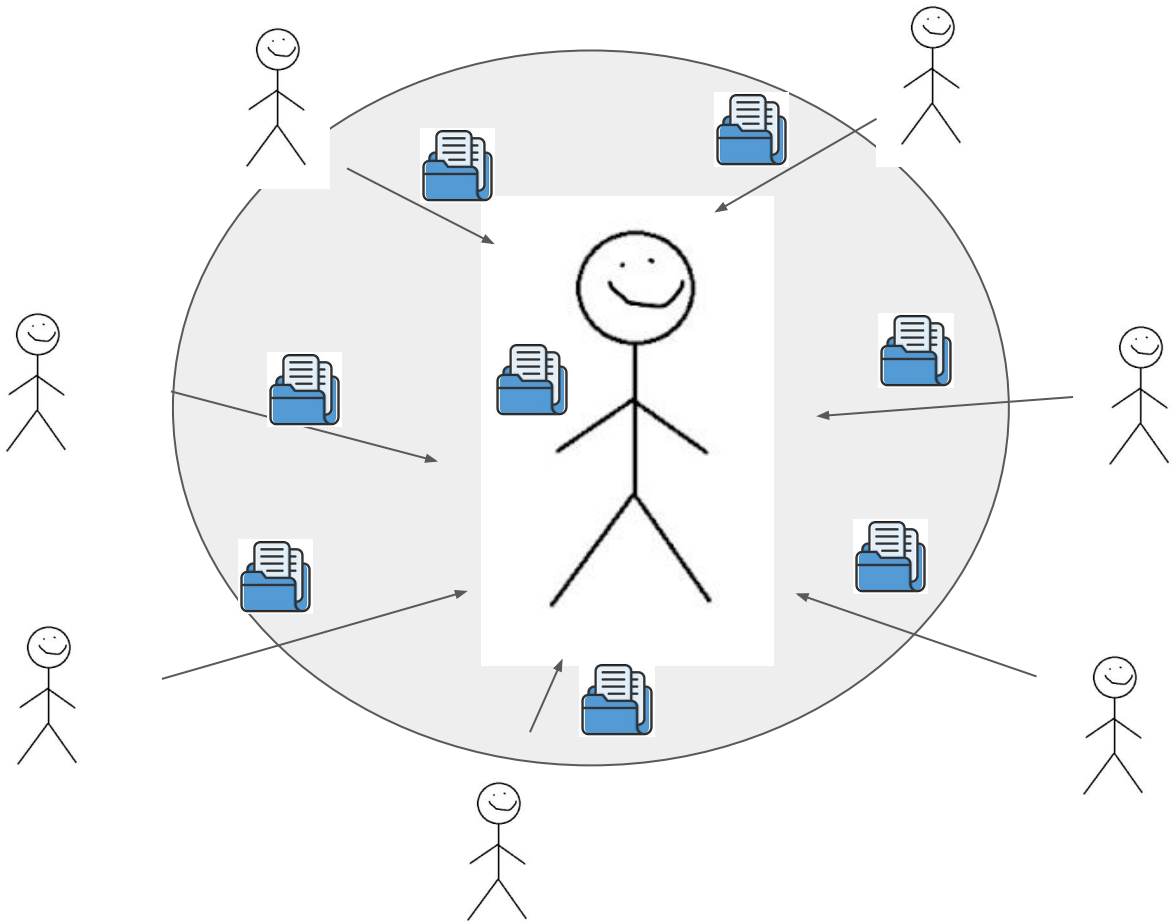
https://matt-welsh.blogspot.com/2009/02/how-i-almost-killed-facebook.html

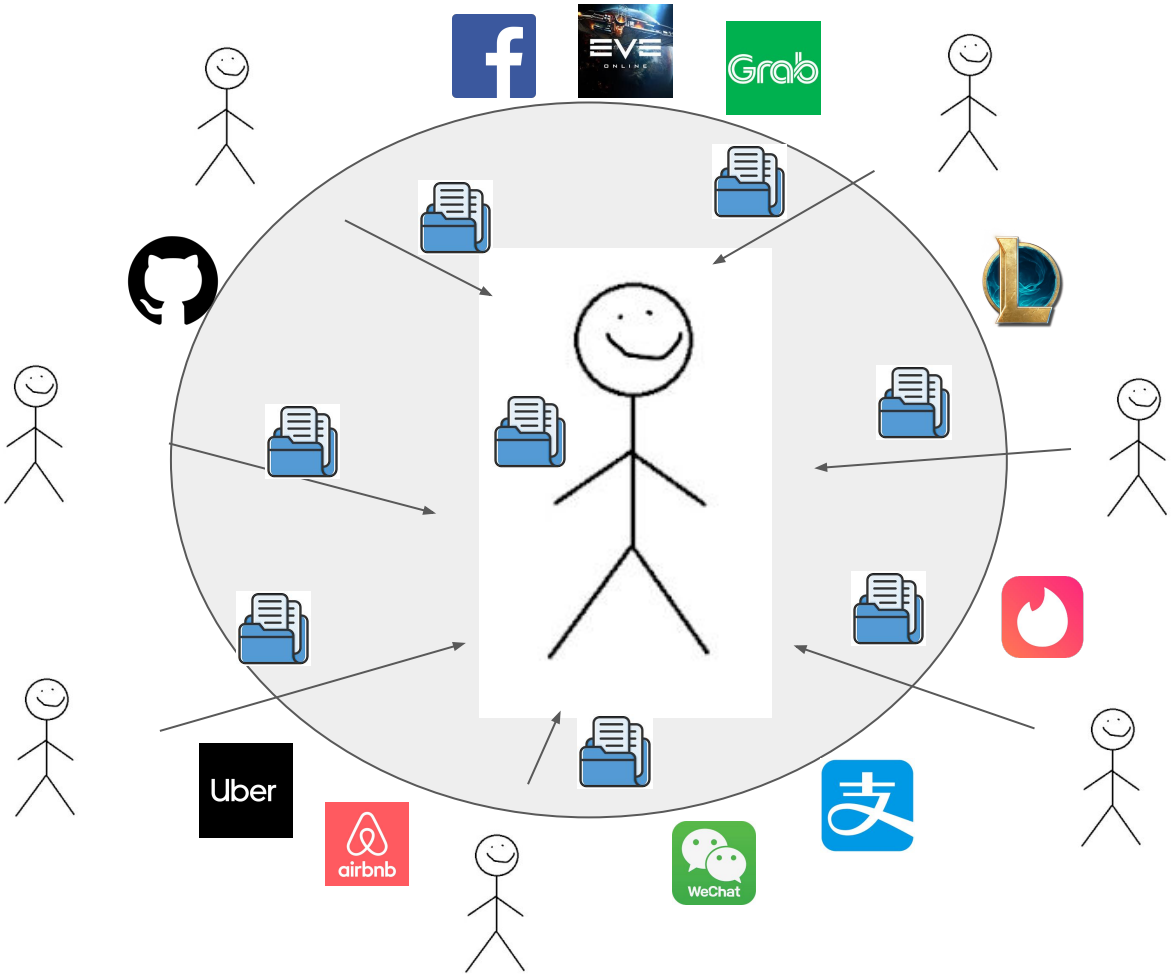



Ps


Ps
X
Grab
WeChat
airbnb
Uber






EVE
ONLINE
Grab
Uber
airbnb
WeChat

Why do servers exist?

Webservers do several things that individuals, and pure p2p networks, can't:

- They coordinate on and perform computations over multiple people's (private) state.
- They provide strong uptime/liveness guarantees.

Choose the problems you care about

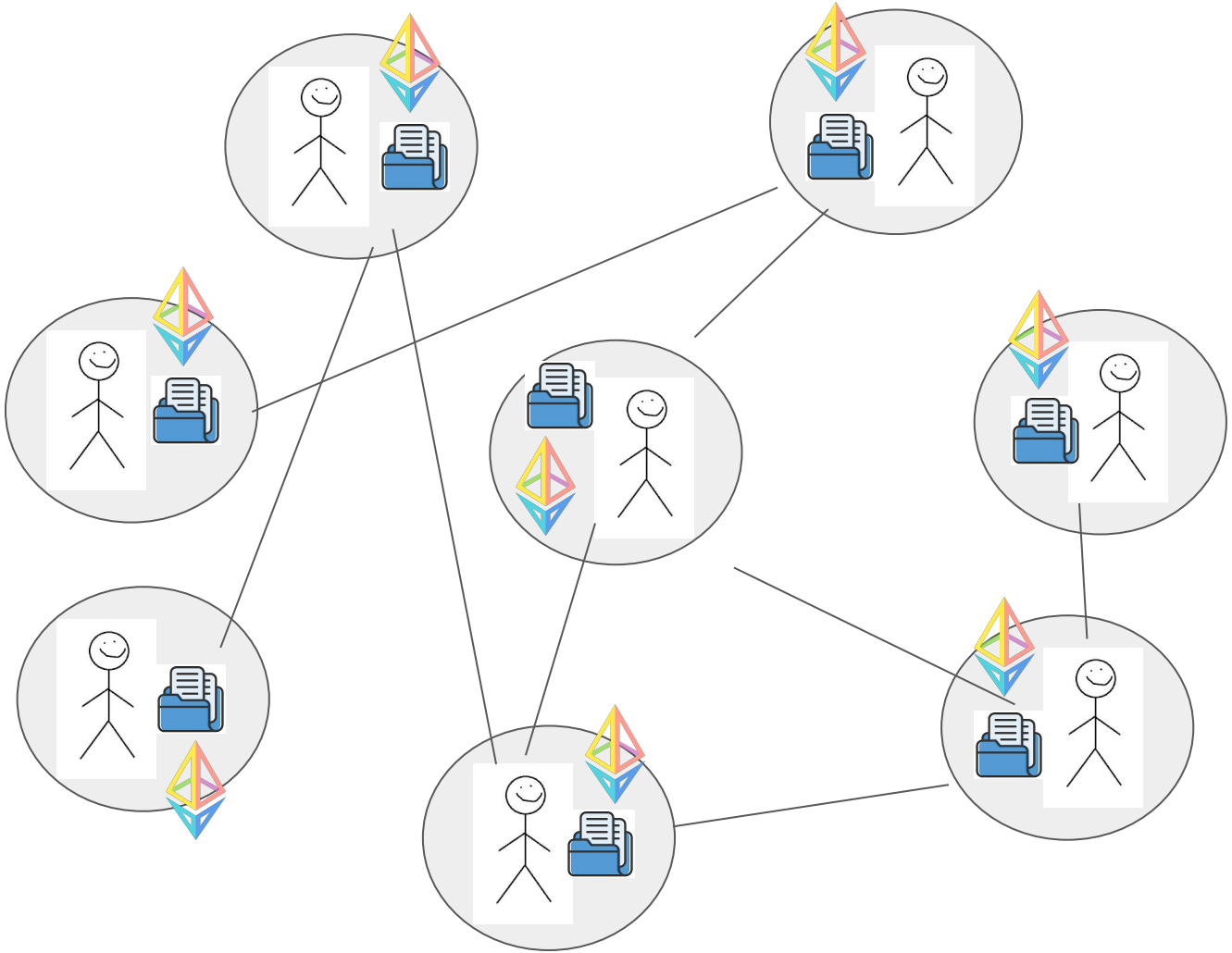
- Vendor lock-in
- Data Silos
- Lack of interoperability
- Single source of failure
- Biases
- Friction monetizing most content
- Poor compute / networking utilization
- Lack of transparency
- Barrier to entry
- Lack of ownership and sovereignty over data
- Etc

How do **we** compute **together**?

How do **we** compute **together**?

Can we do better?

Blockchains, cryptoeconomics, and Ethereum have given us new answers to this question for the first time in decades.



Ethereum has given us...

- Decentralized consensus over global state
- Autonomous and neutral marketplaces & financial derivatives
- Payment systems that no single authority controls
- Permissionless identity registries
- Interoperable and composable games

But we can’t do everything we want yet!

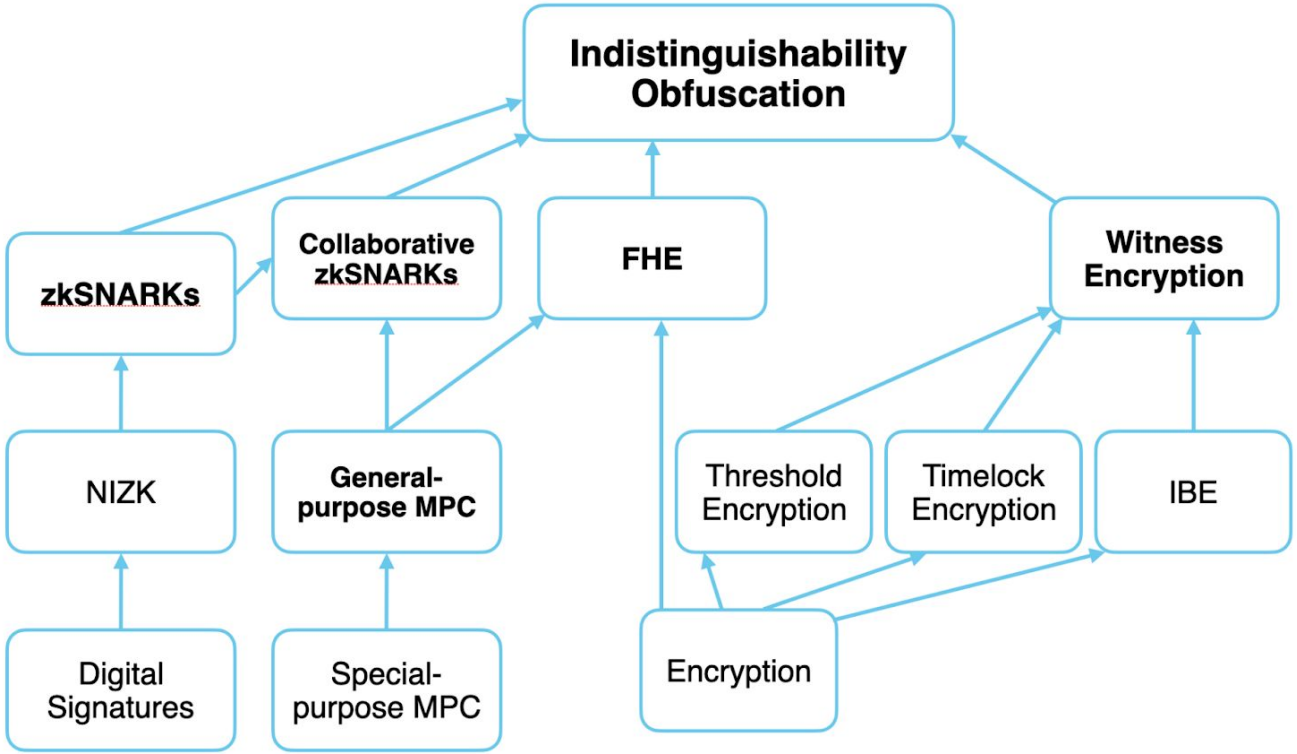
Programmable cryptography can give us many more answers ...

Programmable cryptography can give us many more answers ...

... both independently, and in combination with Ethereum.

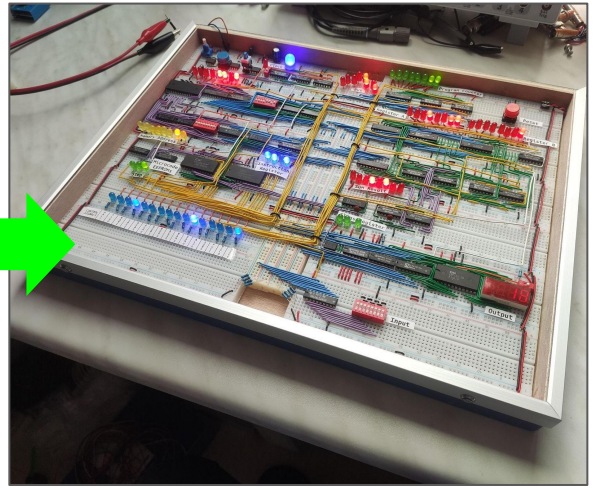
What is **programmable cryptography**?

Programmable Cryptography



Programmable Cryptography

- Proofs for specific functions → proofs for any function
- Verification of specific claims → verification of any claim
- Special-purpose protocols → general-purpose “cryptography compilers”

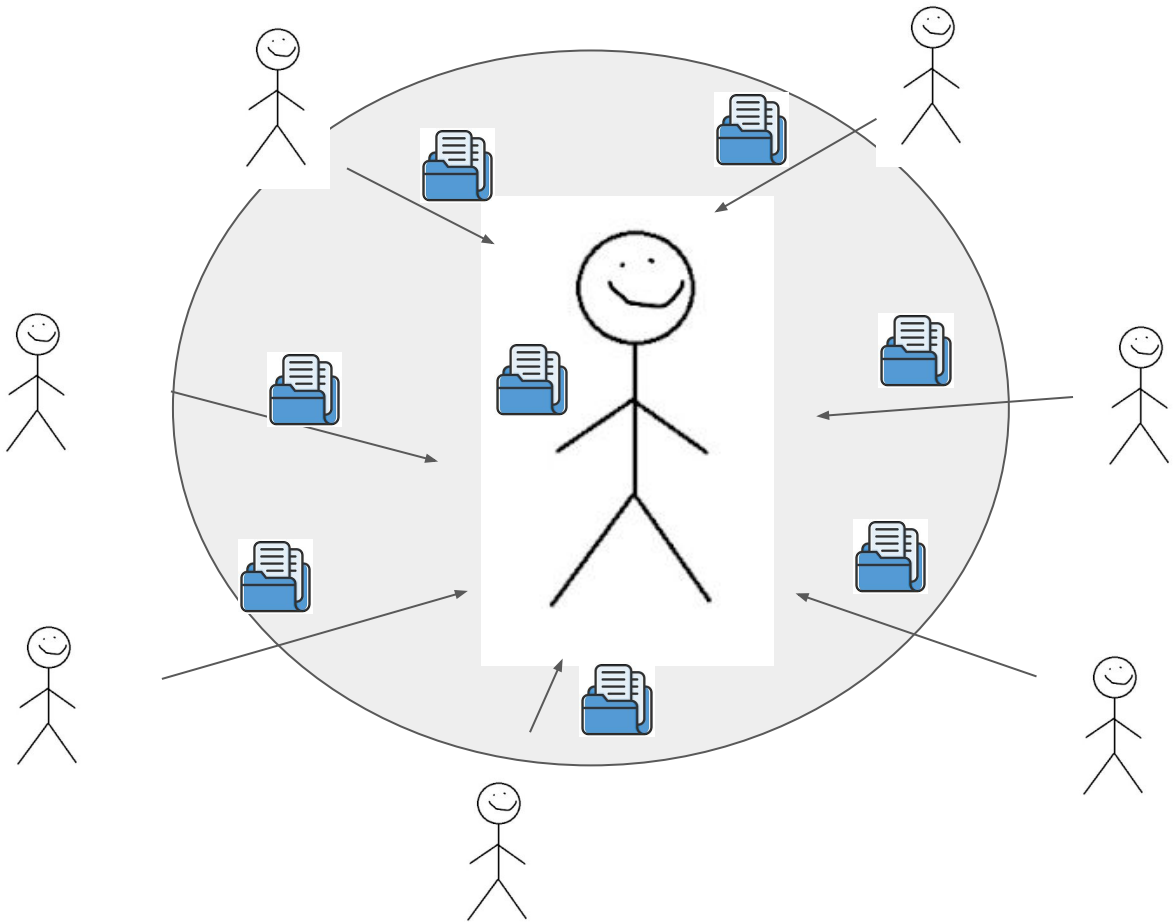


What’s changed?

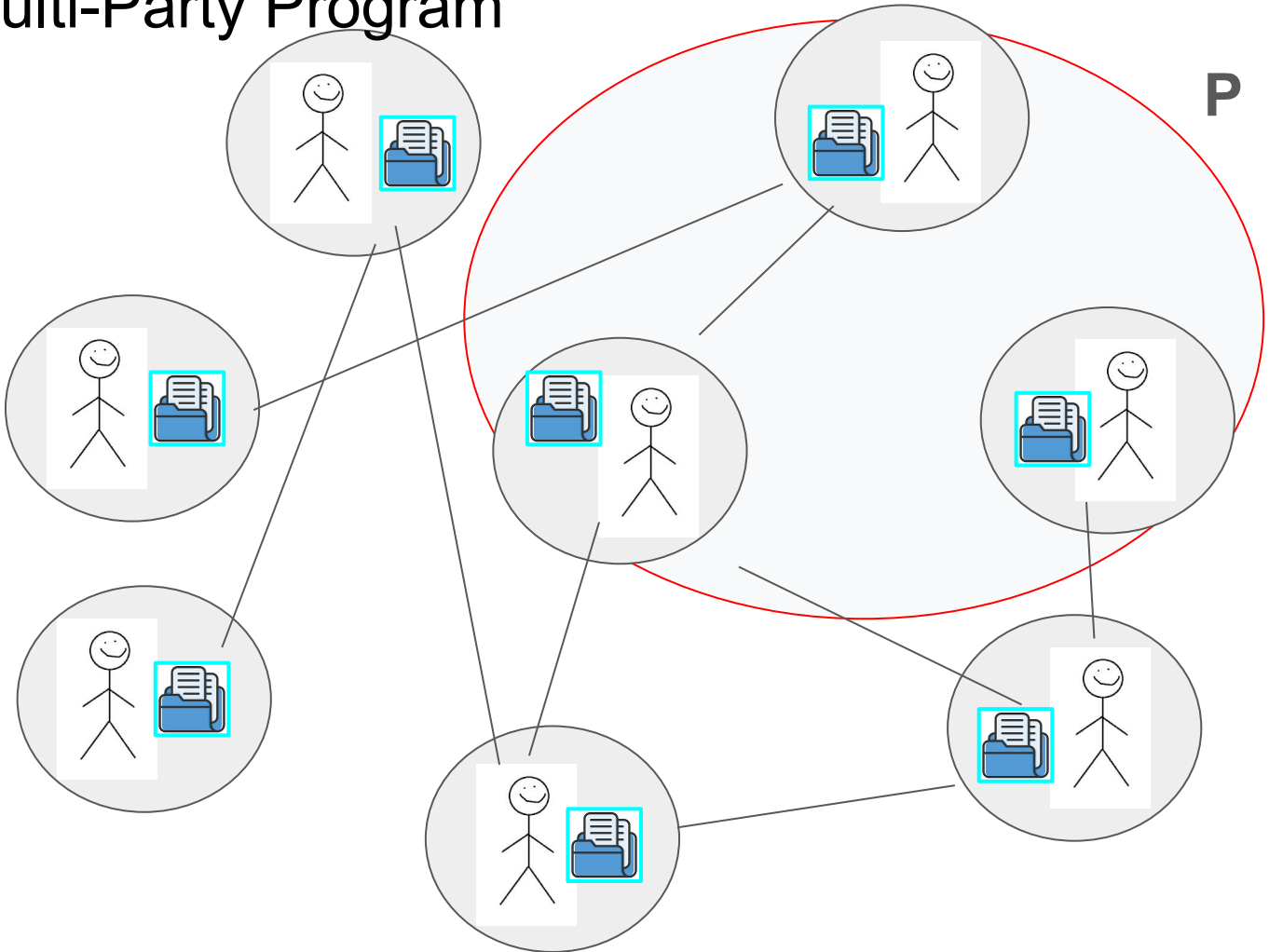
- Significant technological advancements over the last 2-5yr
  - Theoretical level: obfuscation is sound, and candidate schemes are “practical”
  - Engineering level: can develop on top of ZK, FHE in practice; infrastructure exists
- Significant conceptual advances over the 1-2yr
  - Recognized as a unified field that gives a unified set of capabilities
  - We can leverage the power of PC to build extremely powerful systems that were not possible before

How do **we** compute **together**?

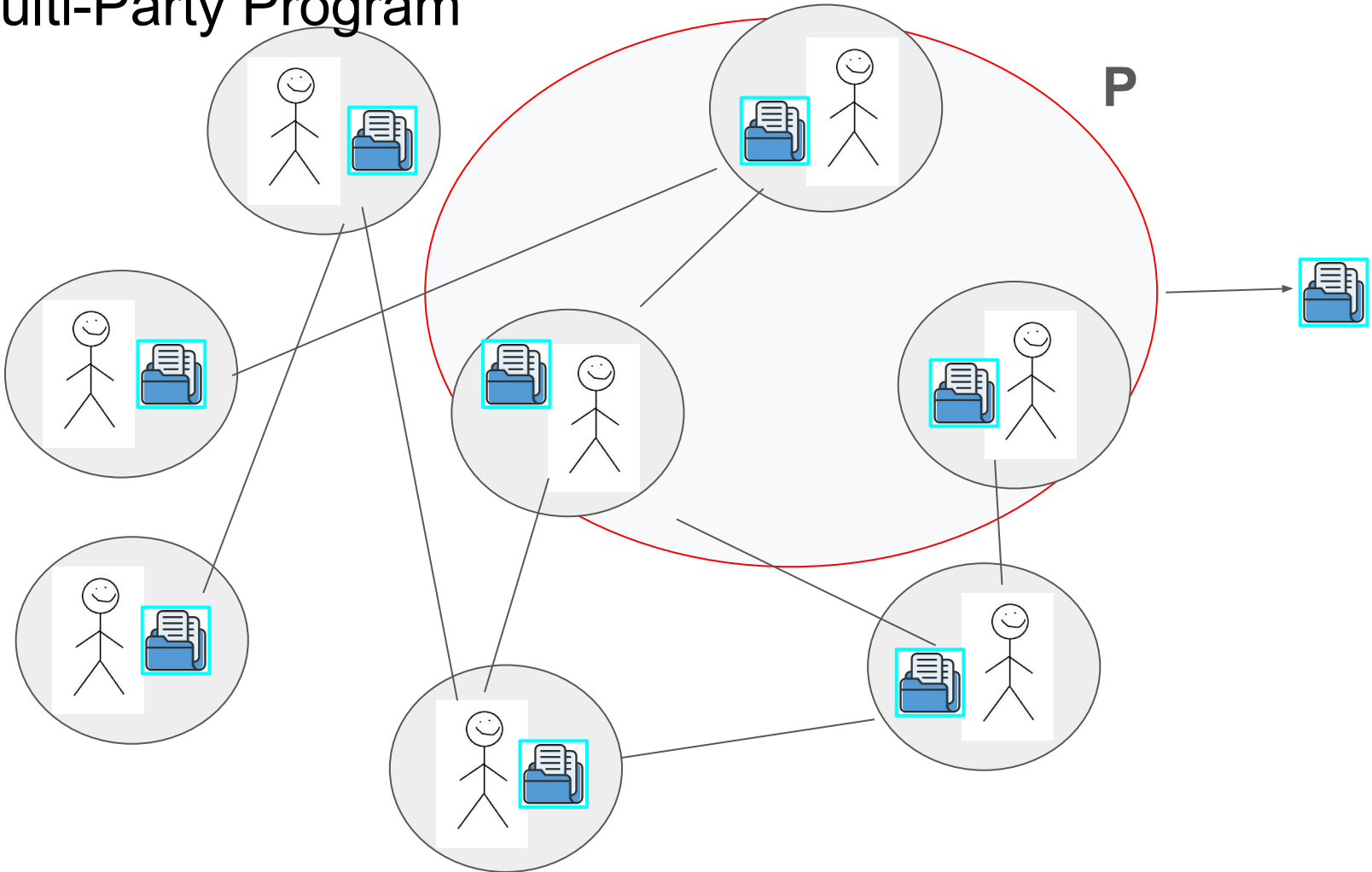
Multi-Party Programming

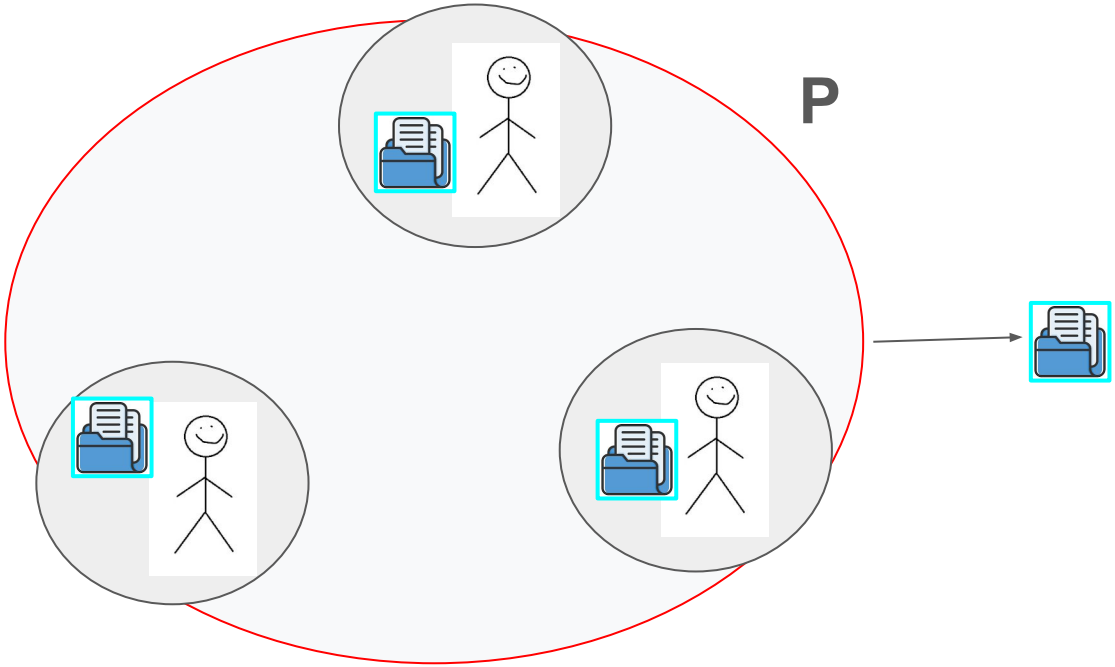


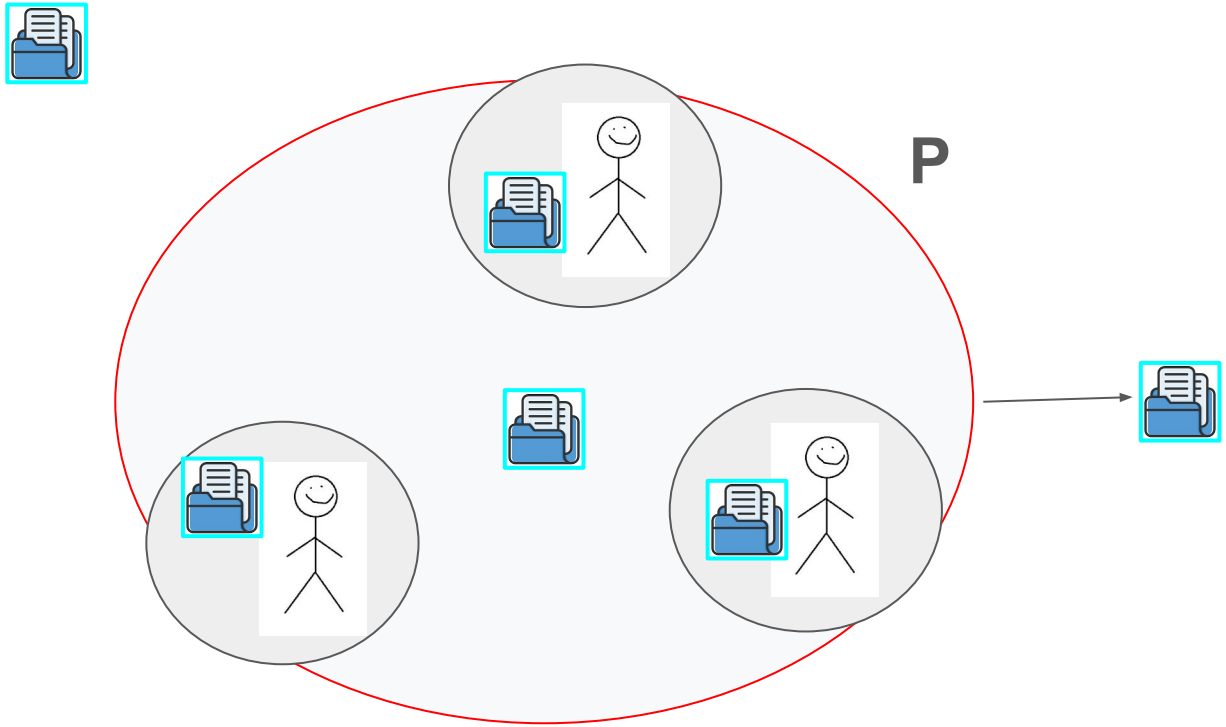
The Multi-Party Program

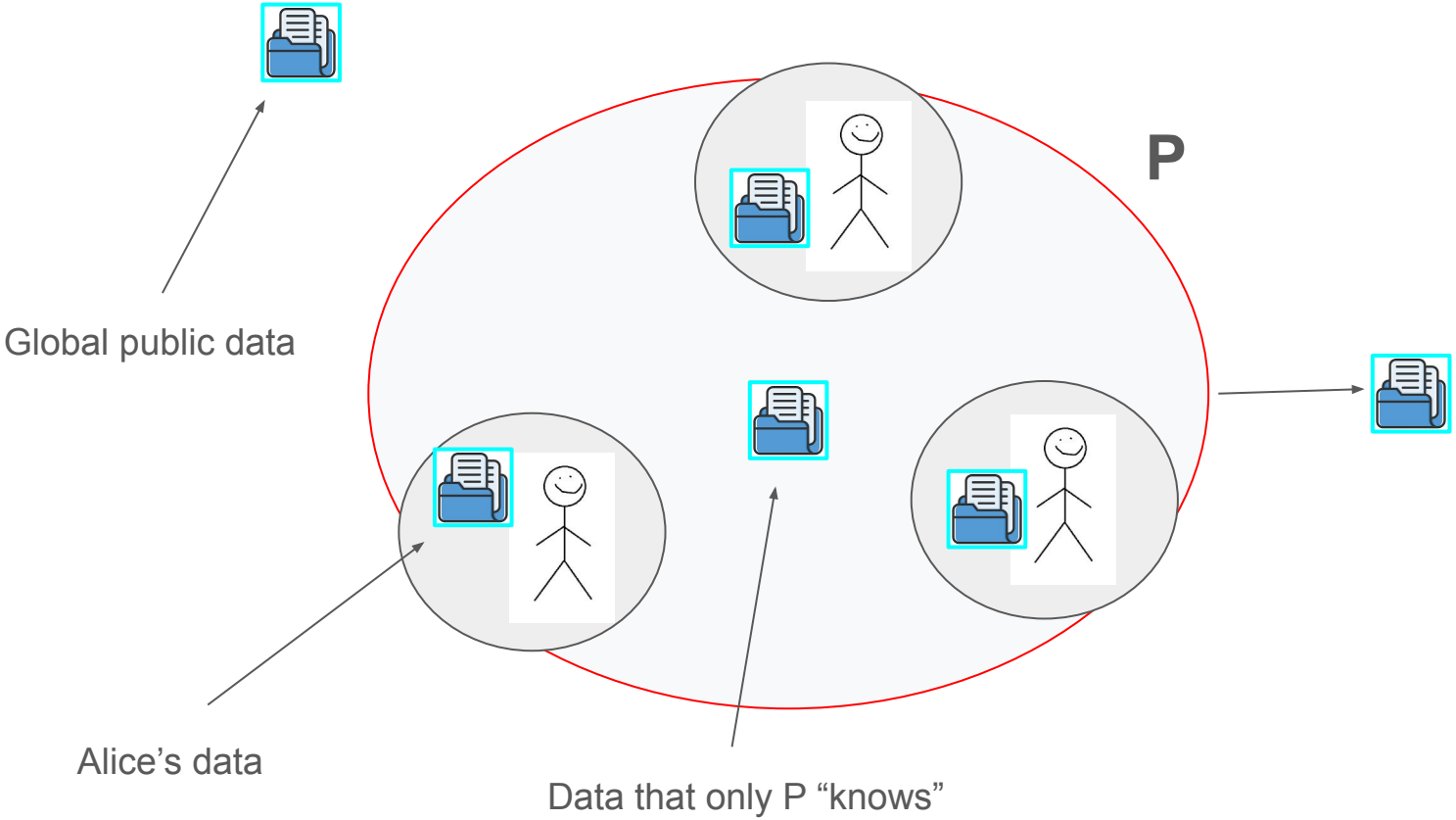


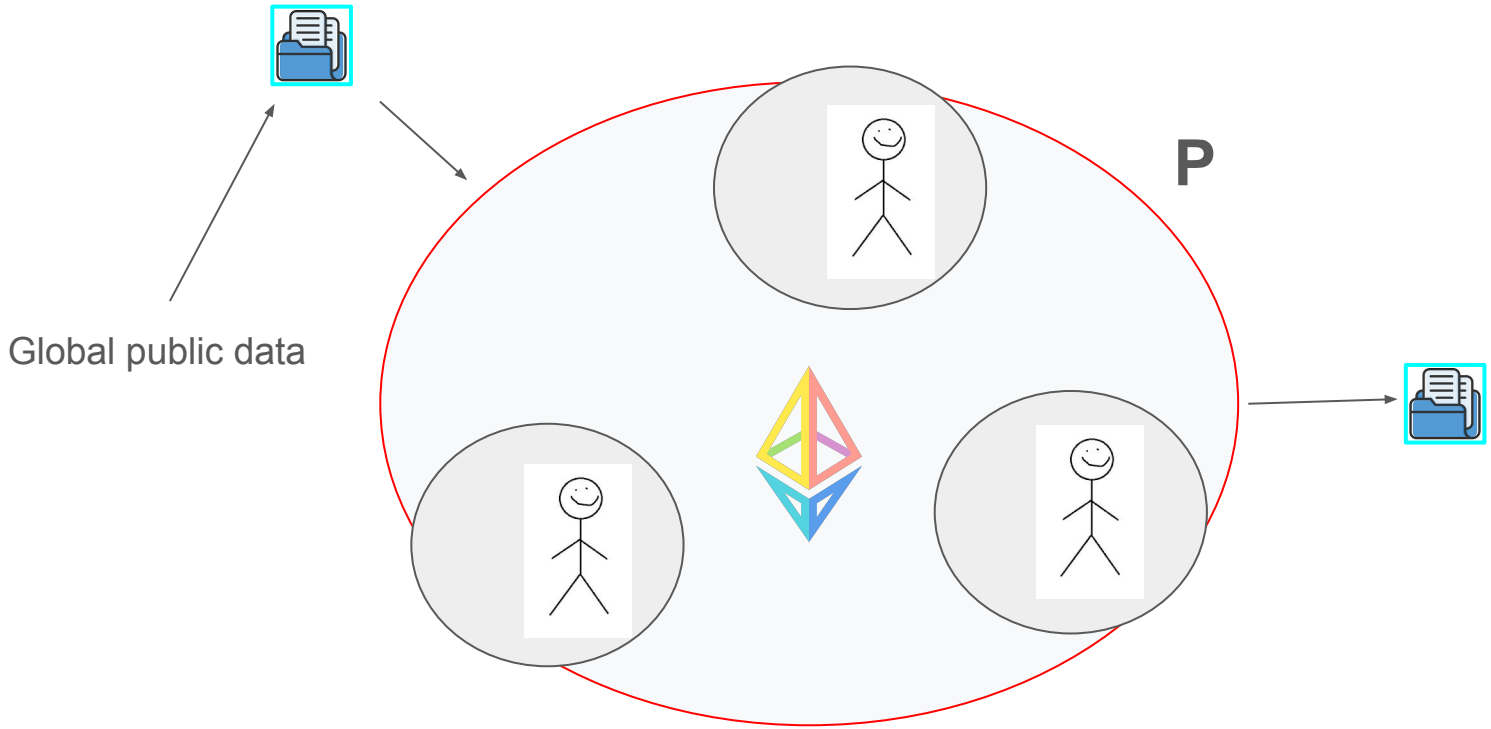
The Multi-Party Program




P


P


P
Global public data
Alice's data
Data that only P "knows"


Global public data
P

Historically, “full” multi-party programming has not been possible with P2P systems or blockchains.

Example: Decentralized Social Media

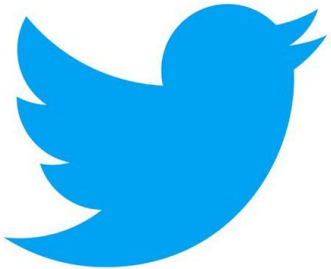
farcaster

Bluesky

NOSTR

Why is everyone trying to build decentralized Twitter?

Why not decentralized Facebook?



The state “topology” of Facebook is much more complicated!

The state “topology” of Twitter

- Everyone can see everything
- Everyone broadcasts to everyone


The state “topology” of Twitter

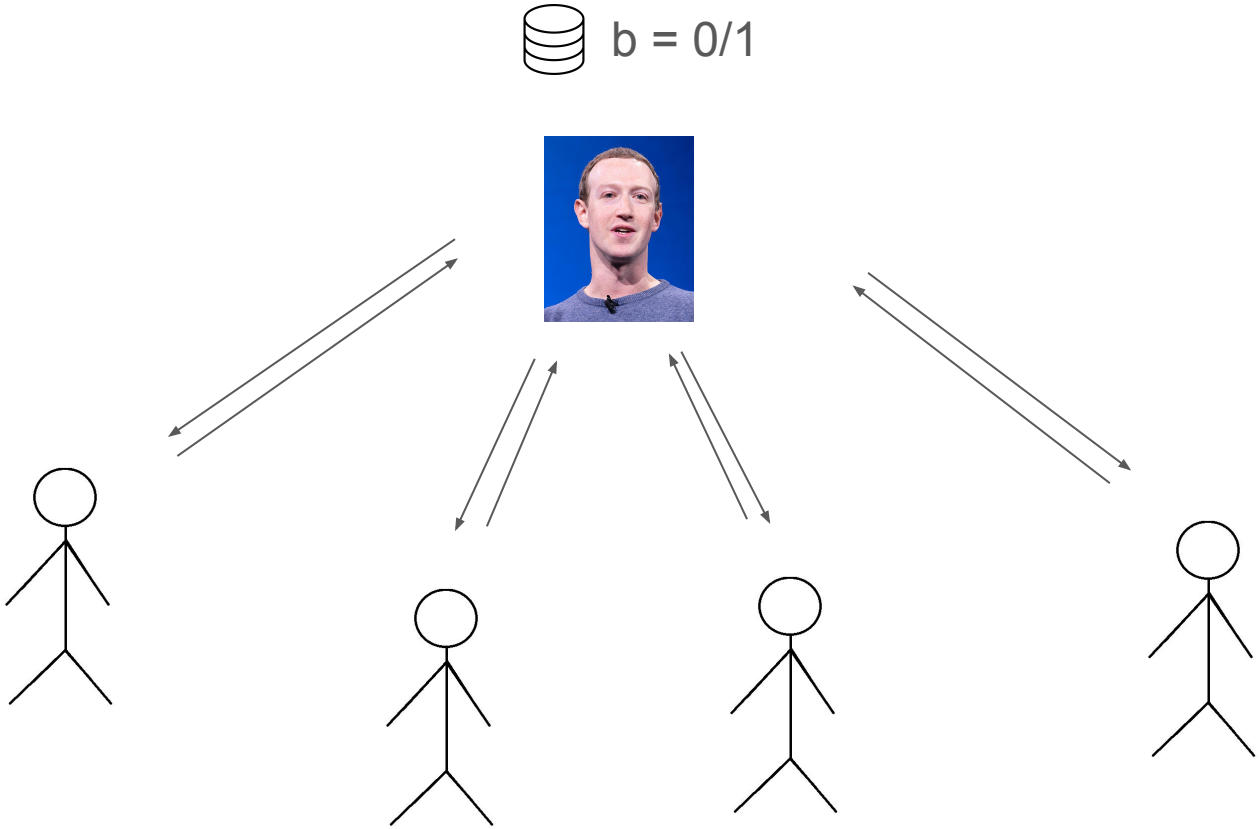
The state “topology” of Facebook

- Friends of friends can see my timeline
- Private groups
- Personalized recommendations based on how your history intersects with other peoples′ histories
- Permissioned APIs whose rules are determined by both your and others′ settings
    - Coefficient of connection
- This person is a 3rd-degree connection

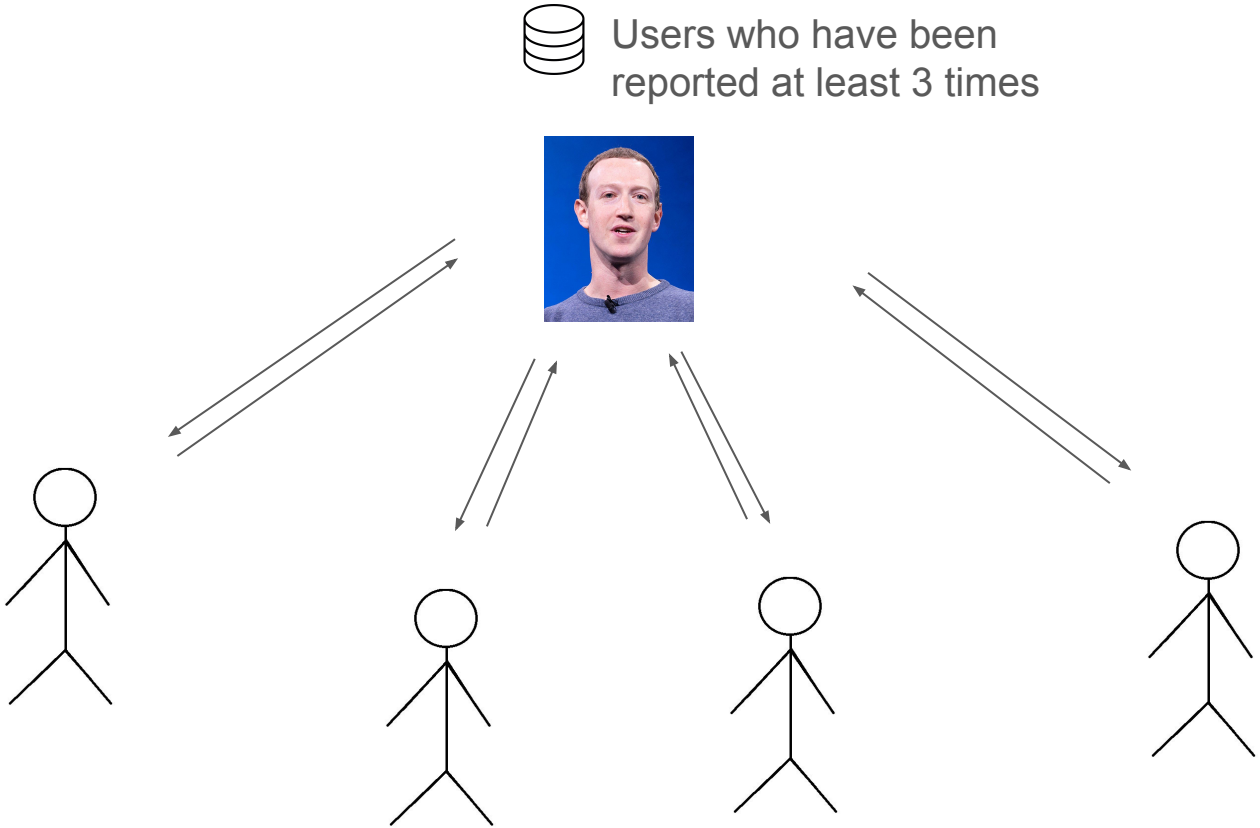
The state “topology” of Facebook

This is impossible to achieve in the general case with only Gen 1 cryptography and consensus.

Some applications have **private global state**



Some applications have **private global state**



Some applications have **private global state**

Something that only the server “knows”

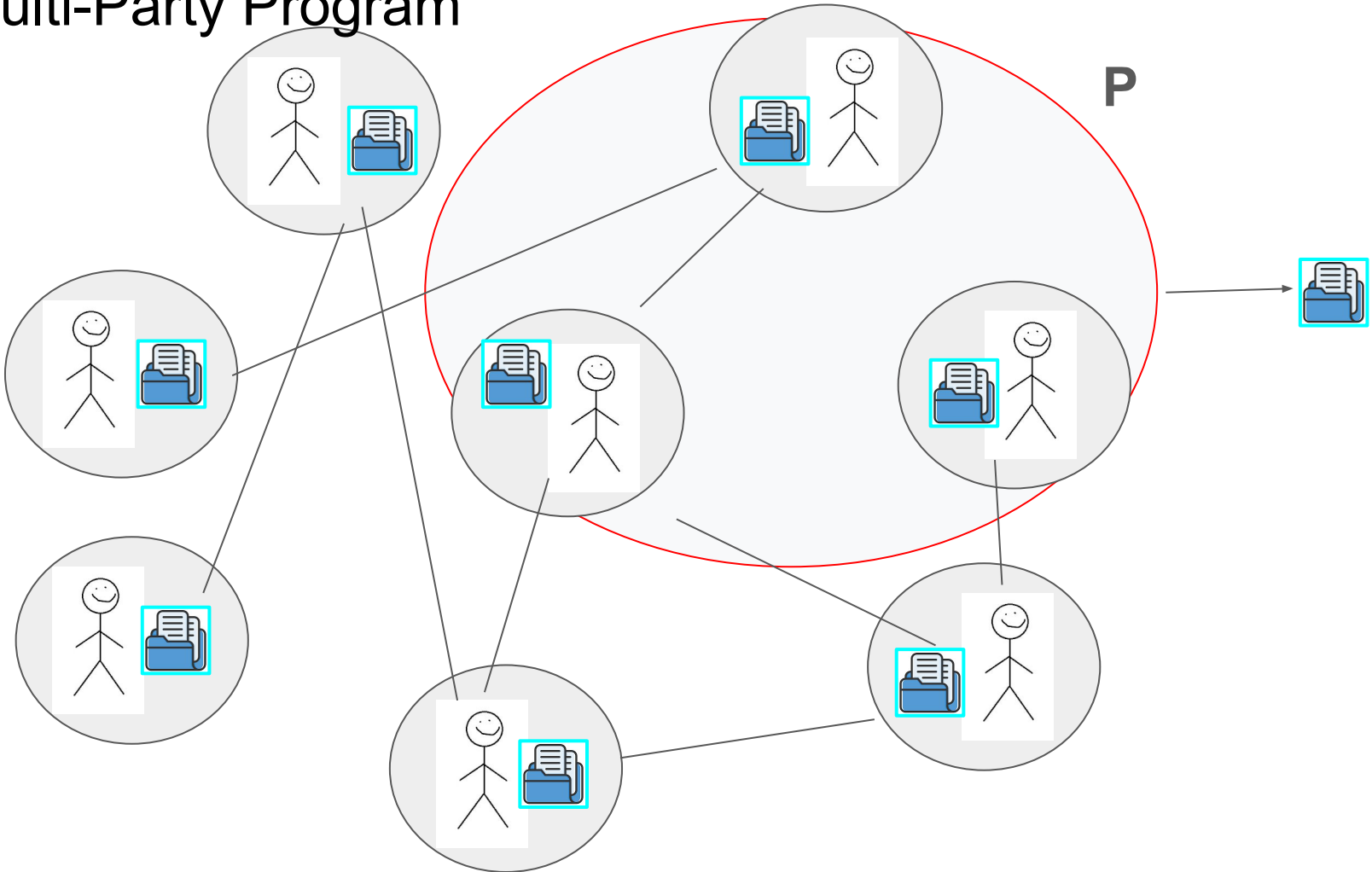
Something that only the server “knows”

Why it’s hard to build decentralized Facebook

- If you have a transparent data layer, individuals can’t have their own state.
  - Decentralized systems generally need their state to be public so it can be verified.
- The system itself can’t have “global private state.”
  - How can you have state that you can operate on, but that doesn’t live anywhere?

How would we build this with programmable cryptography?

The Multi-Party Program

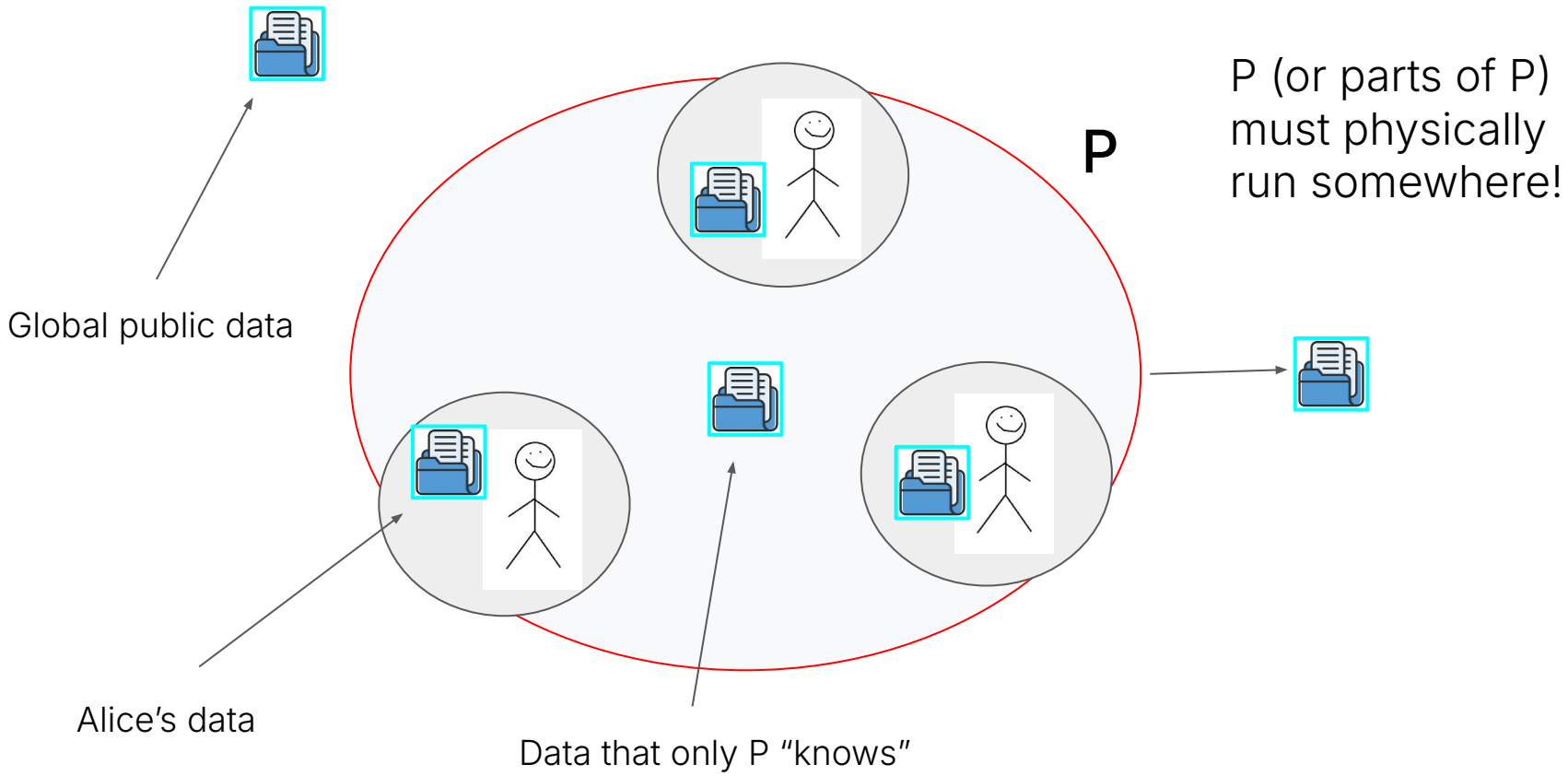


What do we need?

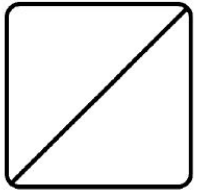
- **Verifiable Computation:** Rather than one trusted actor running the program, multiple (mutually-distrusting) actors might run it, or run different parts.
- **Execution on private state:** Actors can run programs on data they might not know.
- **Data interoperability:** Outputs from one program should be able to be taken as inputs into another program.
- **Consensus:** Everyone agrees on the canonical program state, even without strong liveness assumptions.
- **Non-interactivity:** Scaling to billions of participants.

What do we need?

Verifiable Computation: zkSNARKs and ZKVMs.

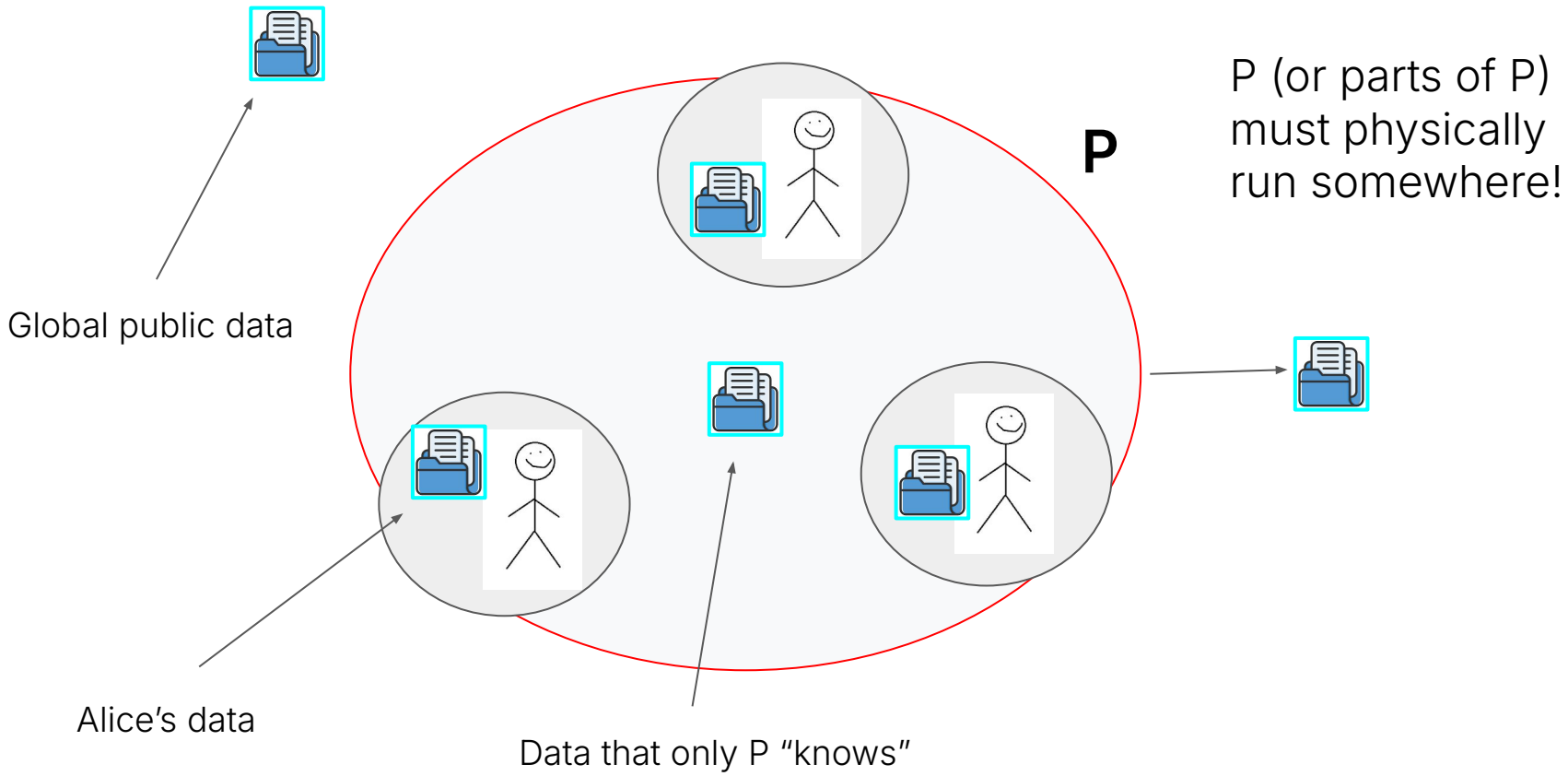


Verifiable Computation: zkSNARKs and ZKVMs.



RISC
ZERO

Programming on Private State: Homomorphic Encryption

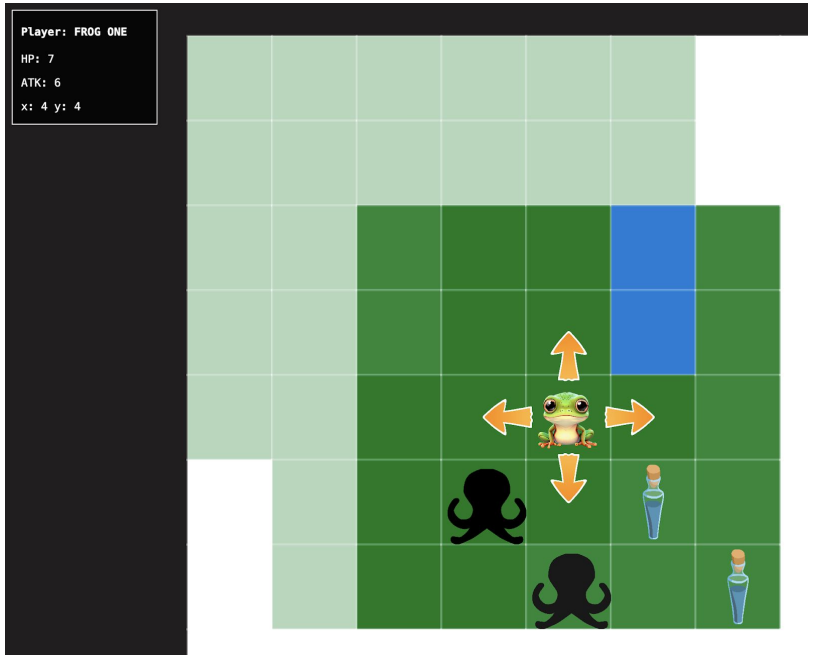


Programming on Private State: Homomorphic Encryption

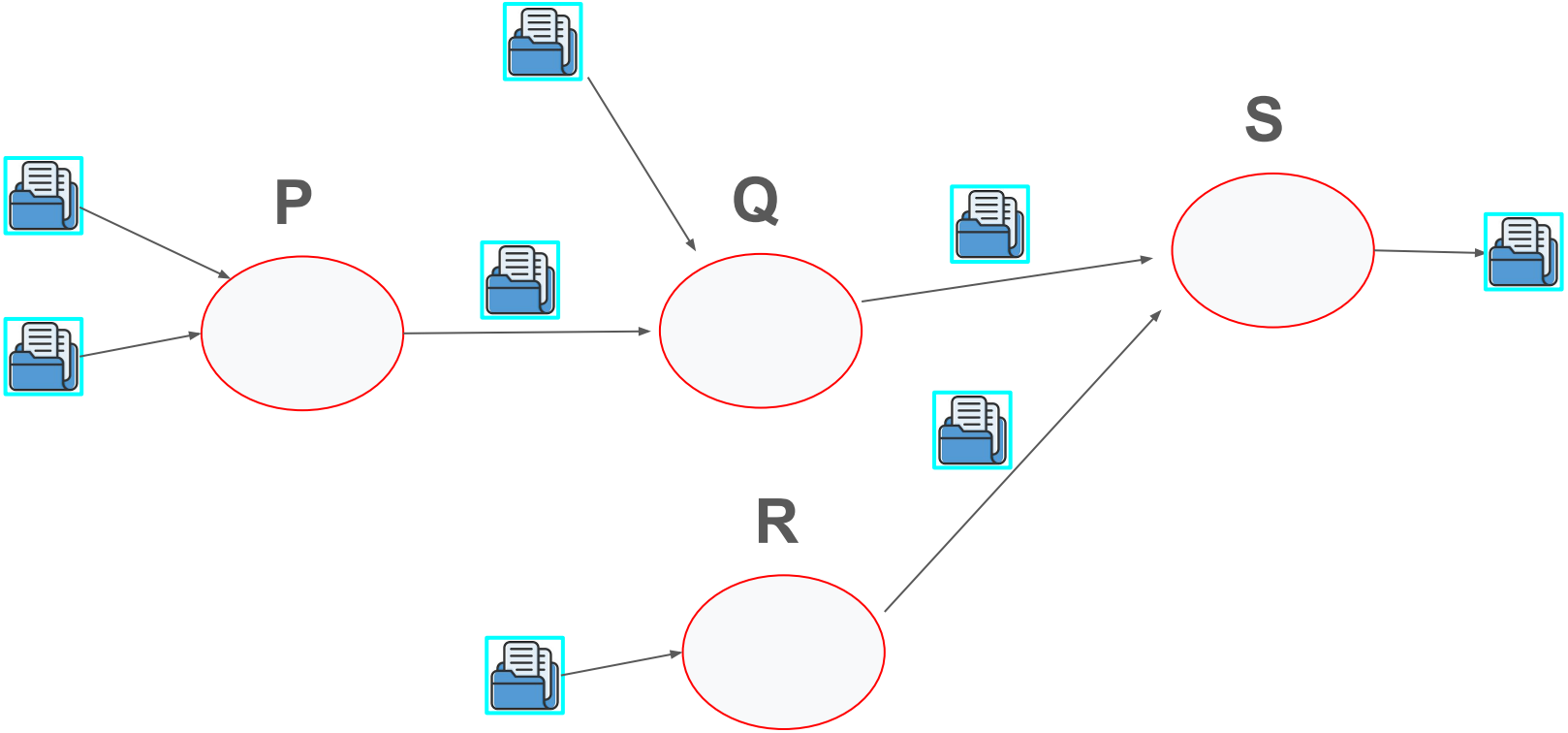
GAUSS phantom-zone

OpenFHE

Open source library for fully homomorphic encryption!

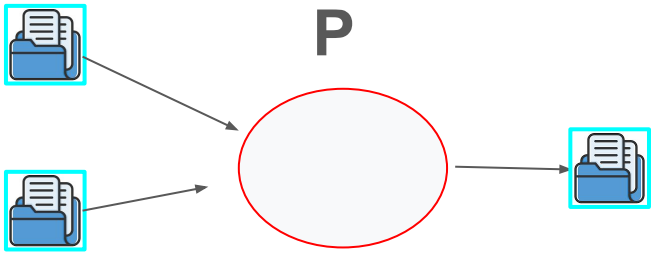


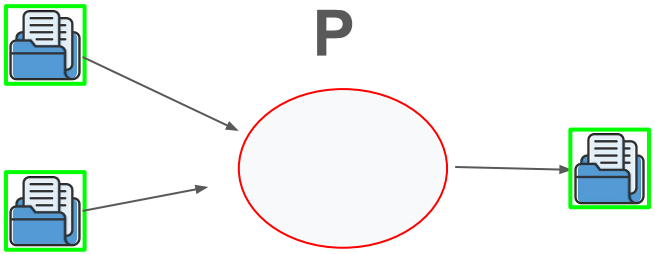
Data Interoperability: Proof-Carrying Data



Data Interoperability: Proof-Carrying Data

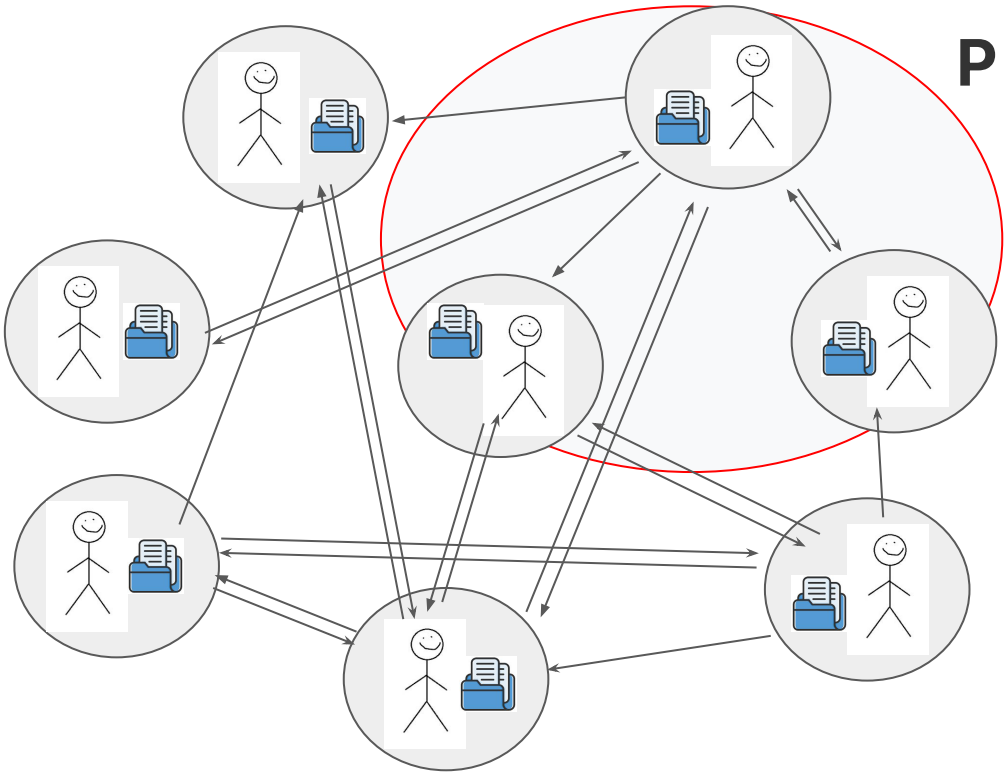
Consensus and Data Availability: Blockchain





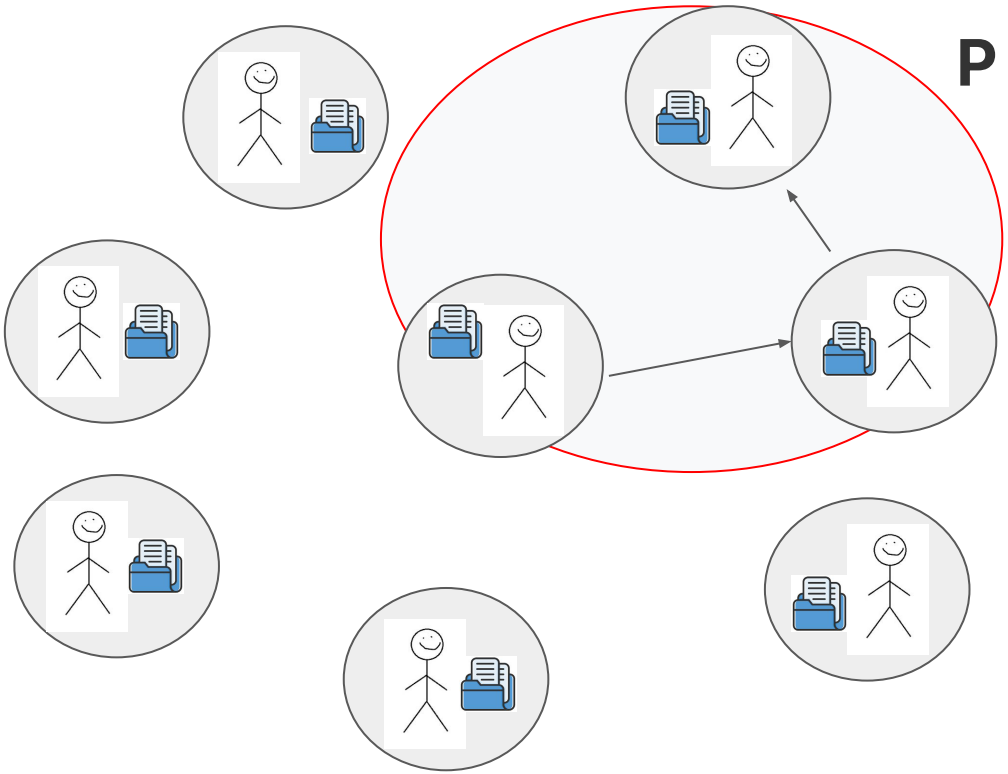
Consensus and Data Availability: Blockchain

Non-Interactivity: Obfuscation and Functional Encryption



Everyone must be online for every interaction

Non-Interactivity: Obfuscation and Functional Encryption



Only the involved parties must be online

Non-Interactivity: Obfuscation and Functional Encryption

Indistinguishability Obfuscation from Well-Founded Assumptions

Aayush Jain* Huijia Lin† Amit Sahai‡

November 12, 2020

Abstract

Indistinguishability obfuscation, introduced by [Barak et. al. Crypto'2001], aims to compile programs into unintelligible ones while preserving functionality. It is a fascinating and powerful object that has been shown to enable a host of new cryptographic goals and beyond. However, constructions of indistinguishability obfuscation have remained elusive, with all other proposals relying on heuristics or newly conjectured hardness assumptions.

In this work, we show how to construct indistinguishability obfuscation from subexponential hardness of four well-founded assumptions. We prove:

Theorem (Informal)**.** Let $\tau \in (0, \infty), \delta \in (0, 1), \epsilon \in (0, 1)$ be arbitrary constants. Assume sub-exponential security of the following assumptions, where λ is a security parameter, p is a λ-bit prime, and the parameters ℓ, k, n are large enough polynomials in λ:

Summary

What can cryptography do, that Ethereum can’t do?

Complex, interoperable state

- How can domains that use different data schemas, proof systems, or semantics talk to each other?
- How can applications hold state with complex predicates on who can see it, and who can operate on it?
- How can an application have state that no one knows?

What can Ethereum do, that
no amount of cryptography can do?

Consensus, Data Availability, and Ordering

- How do we decide what hashes and public keys are *meaningful*?
- How can you prove that something *didn't* happen?
- How can you determine what happened first?
- How can we ensure that data has been made available?
- How can we ensure that participants in a network are live?

What can both do together, that neither can do alone?

What can both do together, that neither can do alone?

- We're increasingly sensing that there's a huge number of yet-to-be discovered, developed, and deployed systems that are unlocked by these fundamentally new capabilities.

How do **we** compute **together**?

A huge design space has opened up of answers that go further than “give all of our stuff to Dave, and have Dave do it for us.”

Let’s uncover it together!

Thank You!

Check out the ProgCrypto Community Hub and Community-Led Session on Friday to learn more about Programmable Cryptography.