DFINITY

# The Internet Computer
## Guest Lecture @ ETH Zurich

December 19, 2022

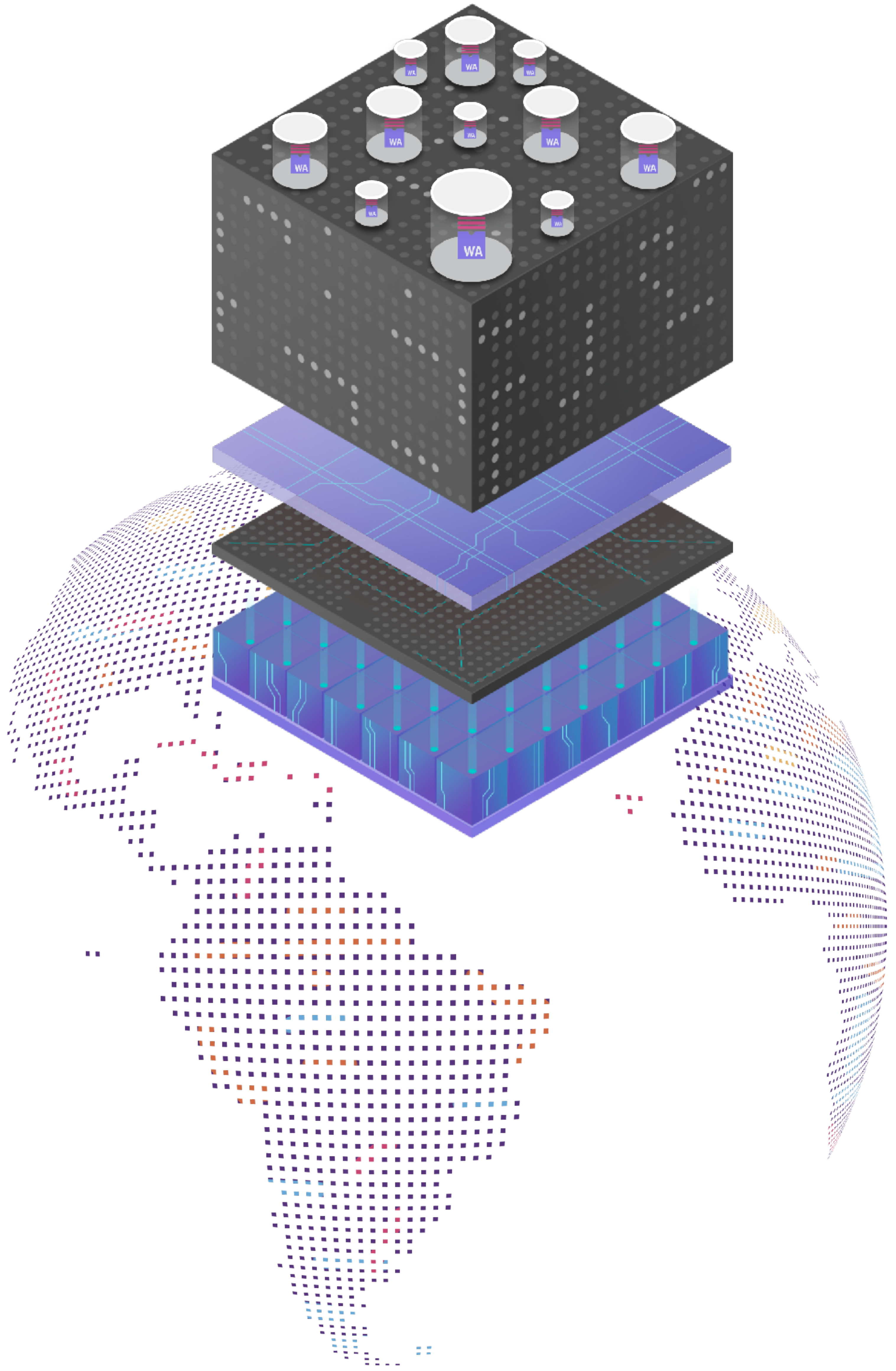
Thomas Locher (thomas.locher@dfinity.org)
Yvonne Anne Pignolet (yvonneanne@dfinity.org)

DFINITY

- Not-for-profit organization that develops the Internet Computer
- Founded in 2016
- Headquarters: Zurich, Switzerland
- Staff: +250

# Outline

- What is the Internet Computer?
- Internet Computer Architecture
- Closer Look: Consensus
- Closer Look: HTTPS Outcalls
- The Internet Computer Today



# What is the Internet Computer?

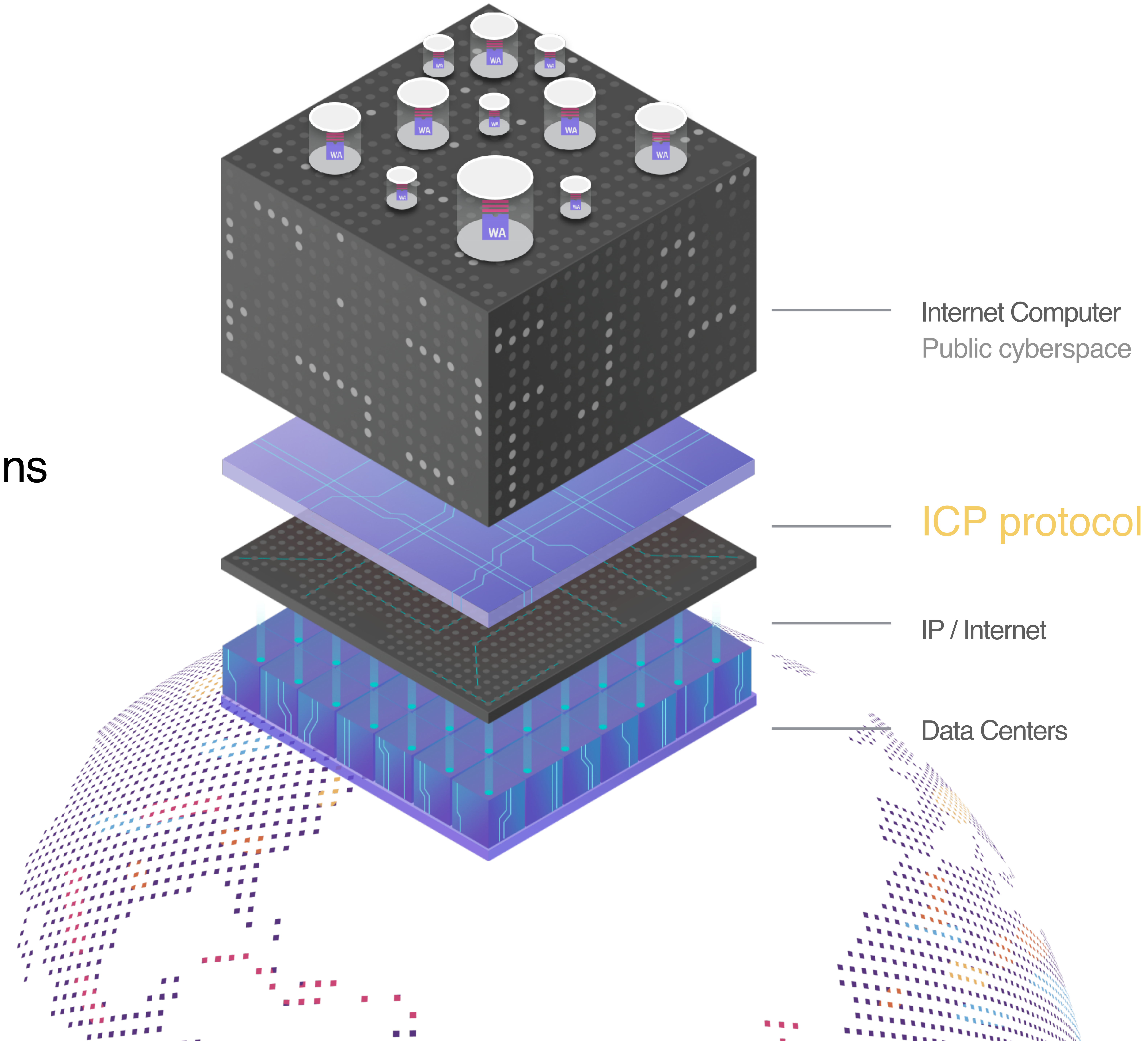
Platform to run **any computation**, using blockchain technology for decentralization and security

# Internet Computer Protocol (ICP)

**Coordination of nodes in independent data centers, jointly performing any computation for anyone**

ICP creates the Internet Computer blockchains

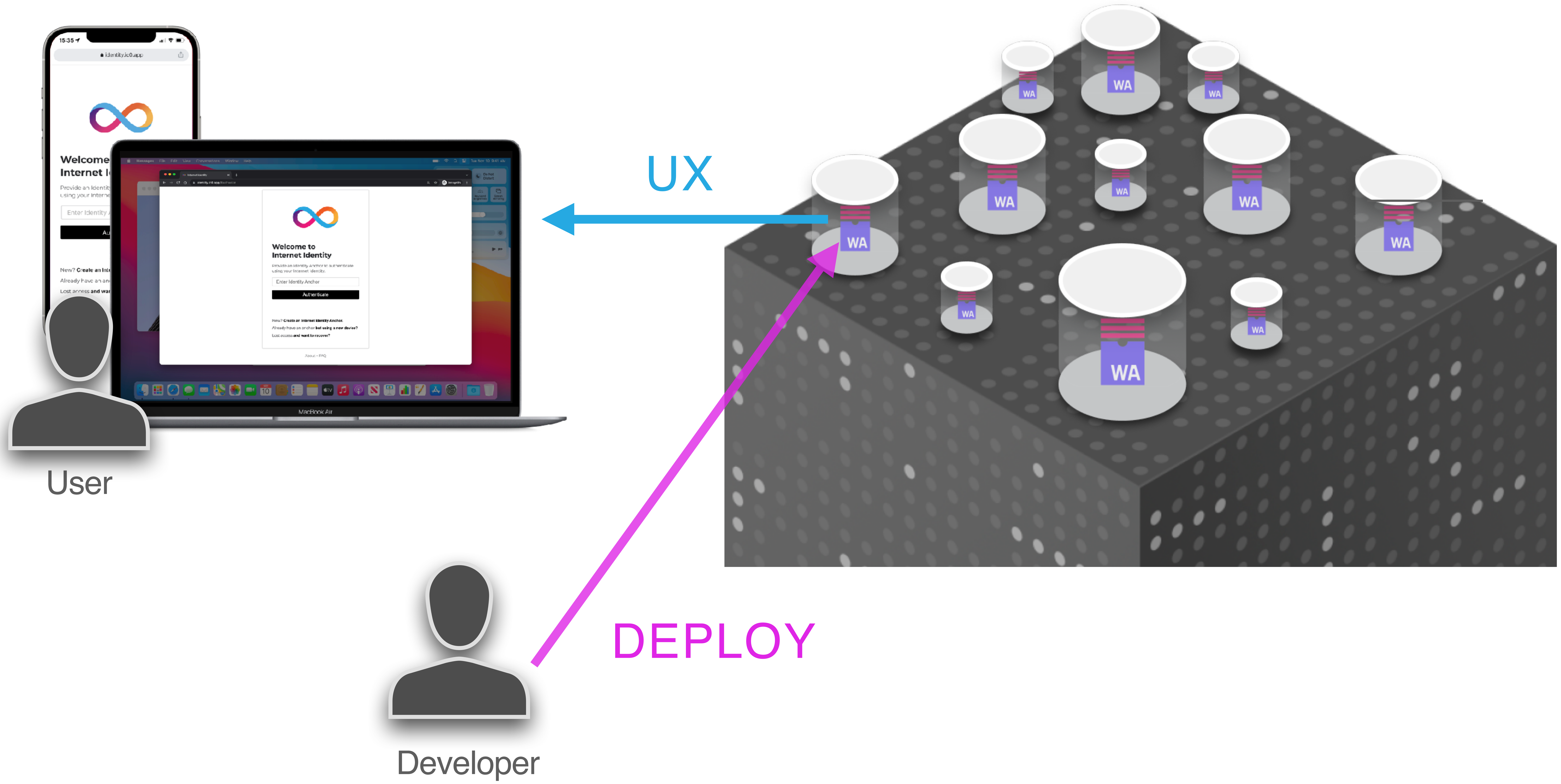
Guarantees safety and liveness of smart contract execution despite Byzantine participants



# Canisters


Data: Memory pages
Canister smart contract
WA
Code: WebAssembly bytecode
WA

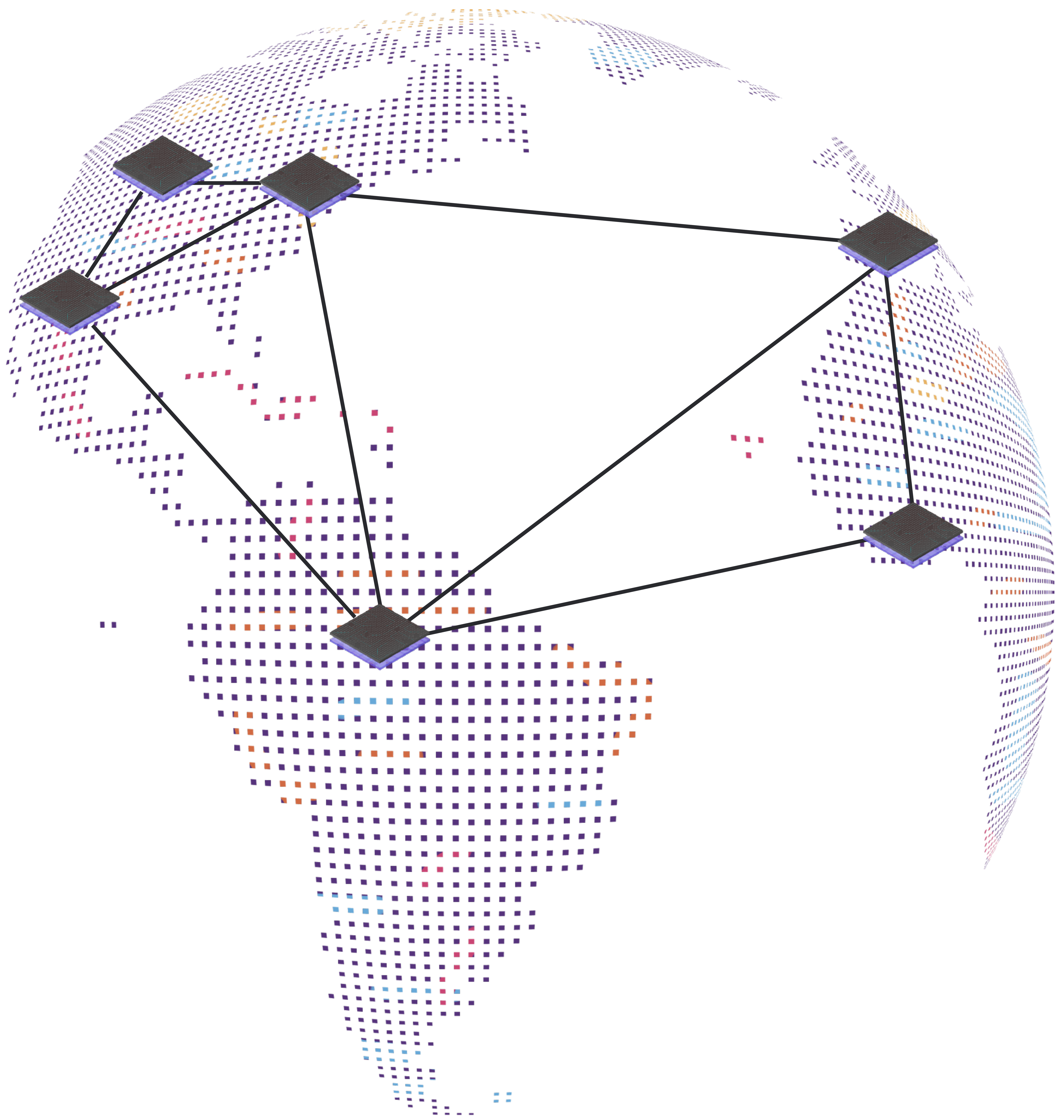
# Deploying and Using Canisters



# Internet Computer Architecture

Collection of **replicated state machines** that are tied together using advanced cryptography

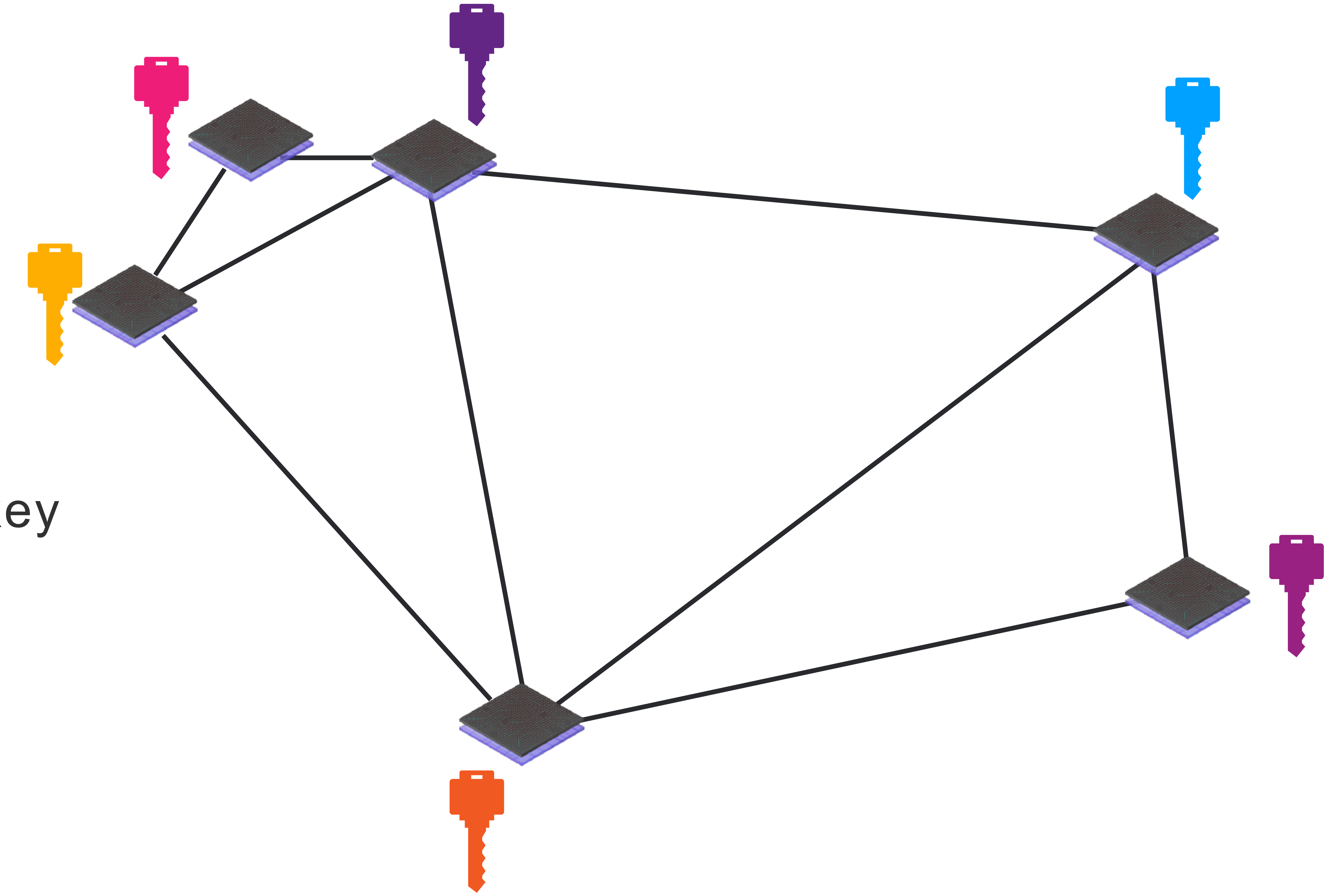
# Nodes in Independent Data Centers

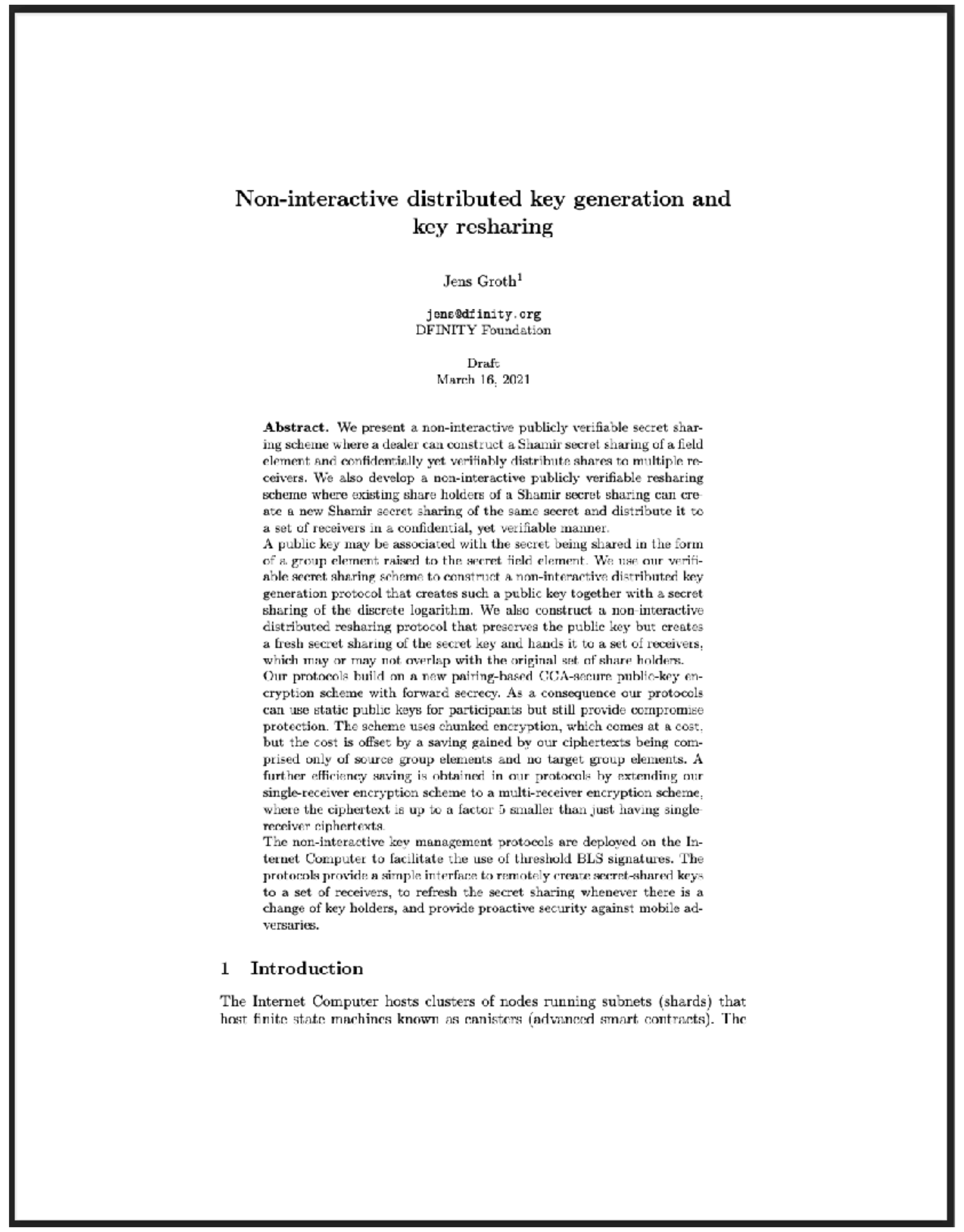


# Chain Key Cryptography

Single 48-byte public key

for a secret-shared private key





**Non-interactive distributed key generation and key resharing**


Jens Groth[1]

jens@dfinity.org
DFINITY Foundation


Draft
March 16, 2021


**Abstract.** We present a non-interactive publicly verifiable secret sharing scheme where a dealer can construct a Shamir secret sharing of a field element and confidentially yet verifiably distribute shares to multiple receivers. We also develop a non-interactive publicly verifiable resharing scheme where existing share holders of a Shamir secret sharing can create a new Shamir secret sharing of the same secret and distribute it to a set of receivers in a confidential, yet verifiable manner.
A public key may be associated with the secret being shared in the form of a group element raised to the secret field element. We use our verifiable secret sharing scheme to construct a non-interactive distributed key generation protocol that creates such a public key together with a secret sharing of the discrete logarithm. We also construct a non-interactive distributed resharing protocol that preserves the public key but creates a fresh secret sharing of the secret key and hands it to a set of receivers, which may or may not overlap with the original set of share holders.
Our protocols build on a new pairing-based CCA-secure public-key encryption scheme with forward secrecy. As a consequence our protocols can use static public keys for participants but still provide compromise protection. The scheme uses chunked encryption, which comes at a cost, but the cost is offset by a saving gained by our ciphertexts being comprised only of source group elements and no target group elements. A further efficiency saving is obtained in our protocols by extending our single-receiver encryption scheme to a multi-receiver encryption scheme, where the ciphertext is up to a factor 5 smaller than just having single-receiver ciphertexts.
The non-interactive key management protocols are deployed on the Internet Computer to facilitate the use of threshold BLS signatures. The protocols provide a simple interface to remotely create secret-shared keys to a set of receivers, to refresh the secret sharing whenever there is a change of key holders, and provide proactive security against mobile adversaries.


**1 Introduction**

The Internet Computer hosts clusters of nodes running subnets (shards) that host finite state machines known as canisters (advanced smart contracts). The

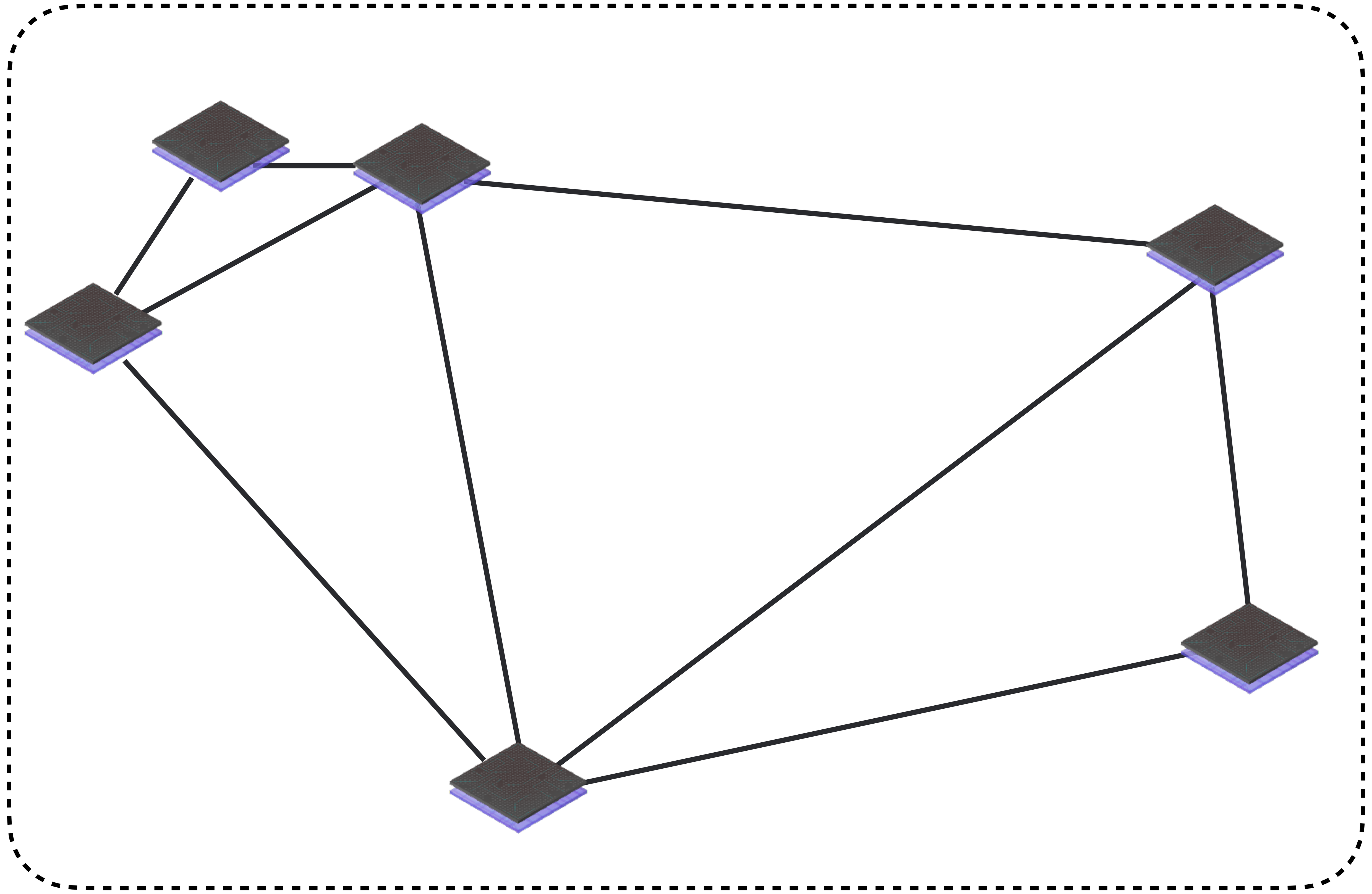
https://internetcomputer.org/how-it-works/chain-key-technology

# Internet Computer Consensus

Assumption: **n > 3f**

Guarantees **agreement** even under asynchrony

Guarantees **termination** under partial synchrony



Internet Computer Consensus

Jan Camenisch, Manu Drijvers, Timo Hanke,
Yvonne-Anne Pignolet, Victor Shoup, Dominic Williams

DFINITY Foundation

May 13, 2021

**Abstract**

We present the Internet Computer Consensus (ICC) family of protocols for atomic broadcast (a.k.a., consensus), which underpin the Byzantine fault-tolerant replicated state machines of the Internet Computer. The ICC protocols are leader-based protocols that assume partial synchrony, and that are fully integrated with a blockchain. The leader changes probabilistically in every round. These protocols are extremely simple and robust: in any round where the leader is corrupt (which itself happens with probability less than 1/3), each ICC protocol will effectively allow another party to take over as leader for that round, with very little fuss, to move the protocol forward to the next round in a timely fashion. Unlike in many other protocols, there are no complicated subprotocols (such as "view change" in PBFT) or unspecified subprotocols (such as "pacemaker" in HotStuff). Moreover, unlike in many other protocols (such as PBFT and HotStuff), the task of reliably disseminating the blocks to all parties is an integral part the protocol, and not left to some other unspecified subprotocol. An additional property enjoyed by the ICC protocols (just like PBFT and HotStuff, and unlike others, such as Tendermint) is *optimistic responsiveness*, which means that when the leader is honest, the protocol will proceed at the pace of the actual network delay, rather than some upper bound on the network delay. We present three different protocols (along with various minor variations on each). One of these protocols (ICC1) is designed to be integrated with a peer-to-peer gossip sub-layer, which reduces the bottleneck created at the leader for disseminating large blocks, a problem that all leader-based protocols, like PBFT and HotStuff, must address, but typically do not. Our Protocol ICC2 addresses the same problem by substituting a low-communication reliable broadcast subprotocol (which may be of independent interest) for the gossip sub-layer.

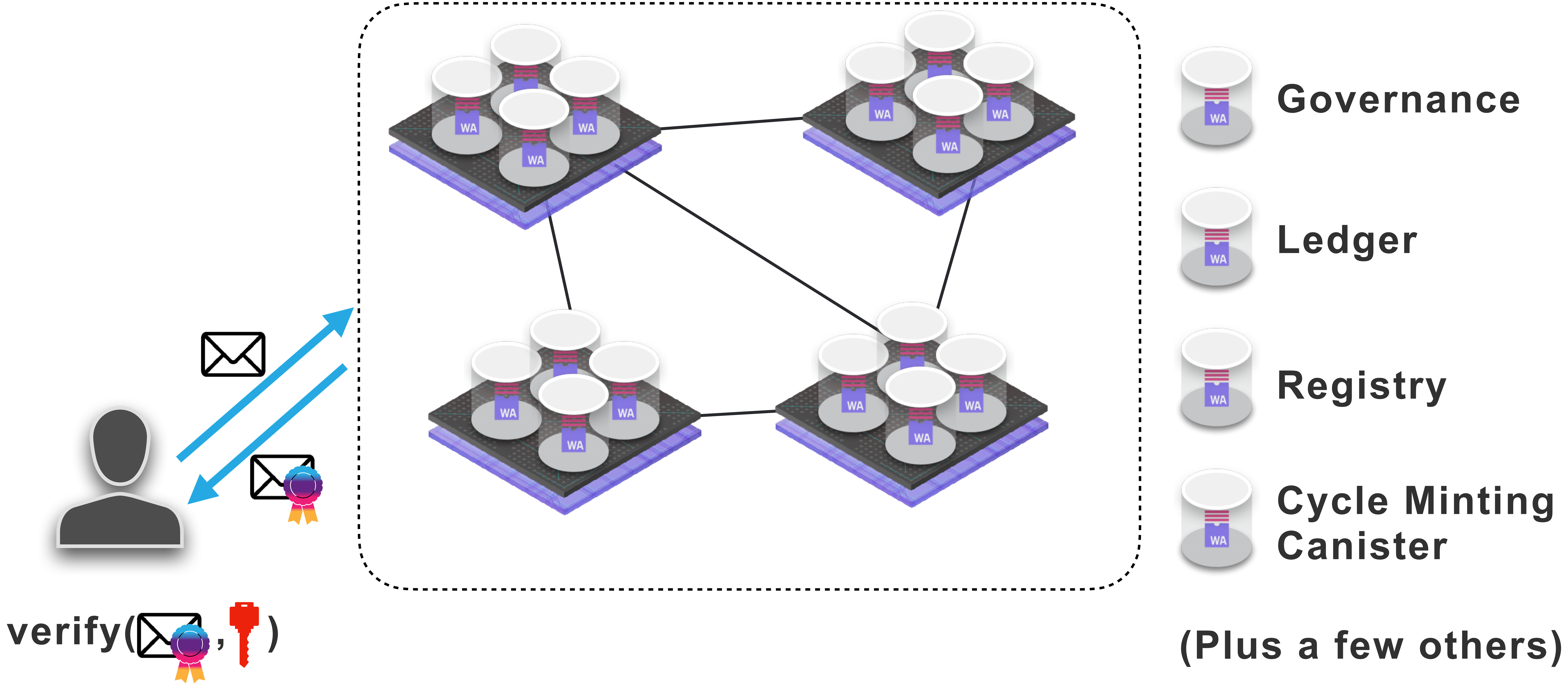
## 1 Introduction

*Byzantine fault tolerance (BFT)* is the ability of a computing system to endure arbitrary (i.e., Byzantine) failures of some of its components while still functioning properly as a whole. One approach to achieving BFT is via *state machine replication* [Sch90]: the logic of the system is replicated across a number of machines, each of which maintains state, and updates its state is by executing a sequence of *commands*. In order to ensure that the non-faulty machines end up in the same state, they must each deterministically execute the same sequence of commands. This is achieved by using a protocol for *atomic broadcast*.

1

https://internetcomputer.org/how-it-works/consensus

# Network Nervous System (NNS)




https://internetcomputer.org/nns

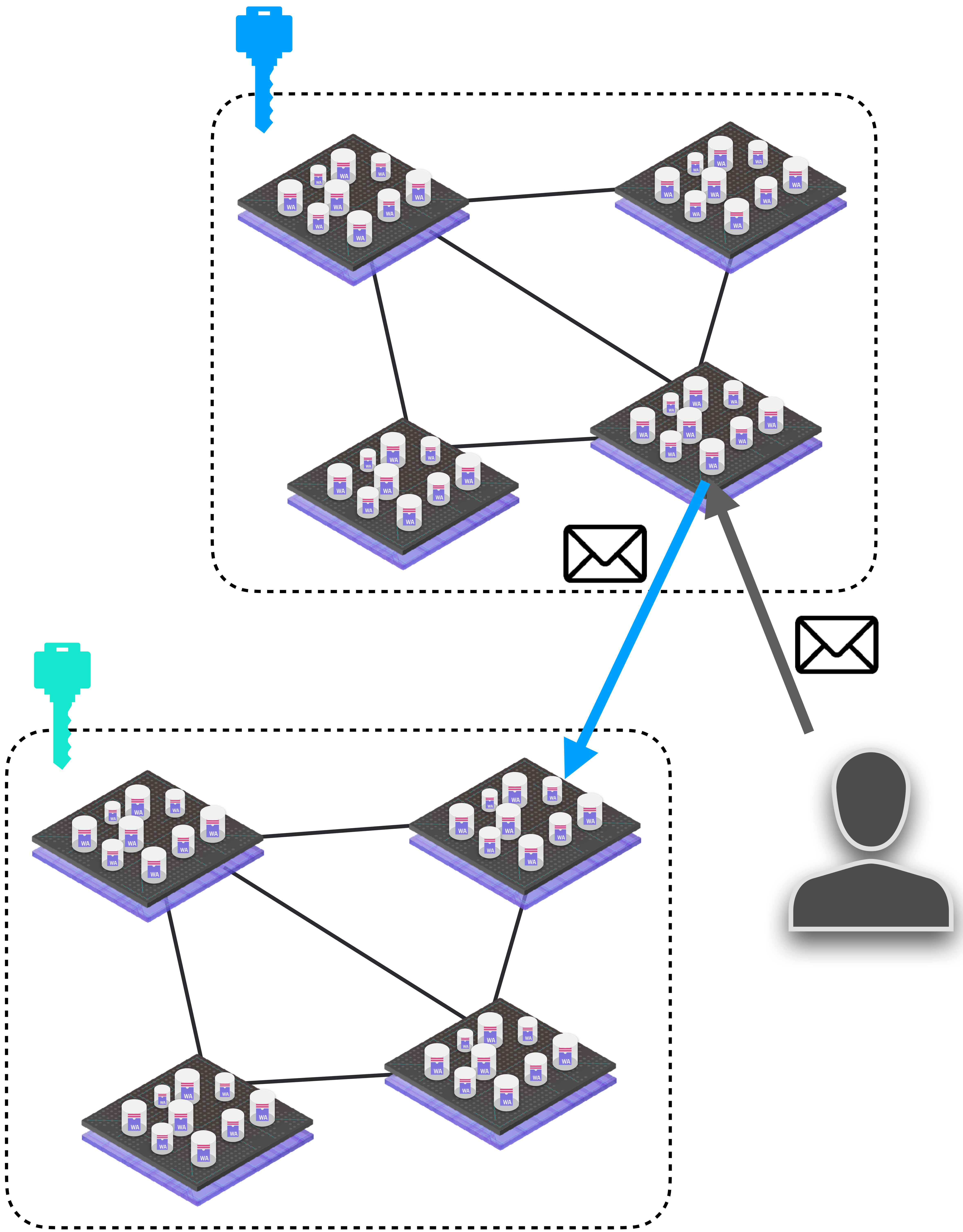
# Subnets


NNS
Registry
Subnet 1: Public key: , nodes: …
Subnet 2: Public key: , nodes: …
…
WA

# Subnets

# Subnets

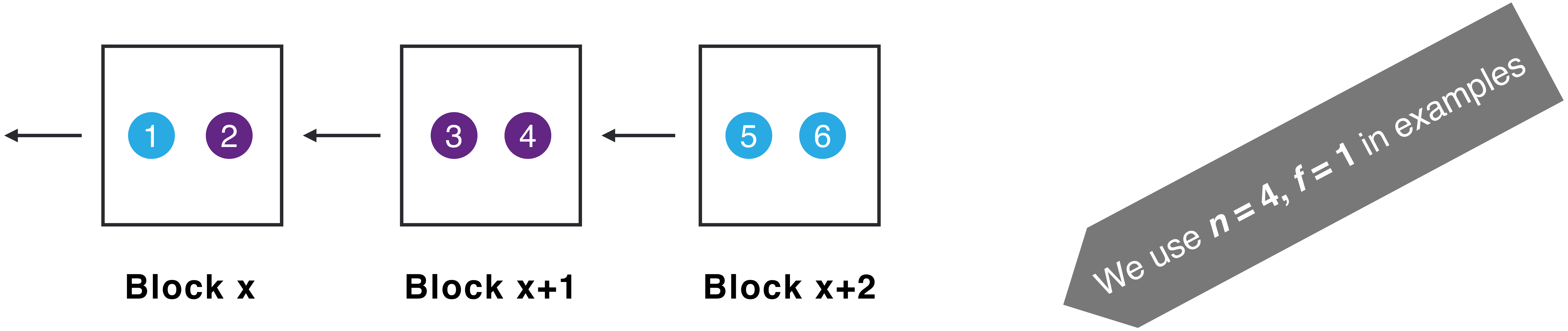
- Each canister is assigned to one subnet
- Each subnet is a **replicated state machine**
- A canister can call canisters on other subnets
- Subnets make the Internet Computer **scalable**!



# Closer Look: Consensus

# Consensus Properties

Messages are placed in **blocks**. We reach agreement using a blockchain.

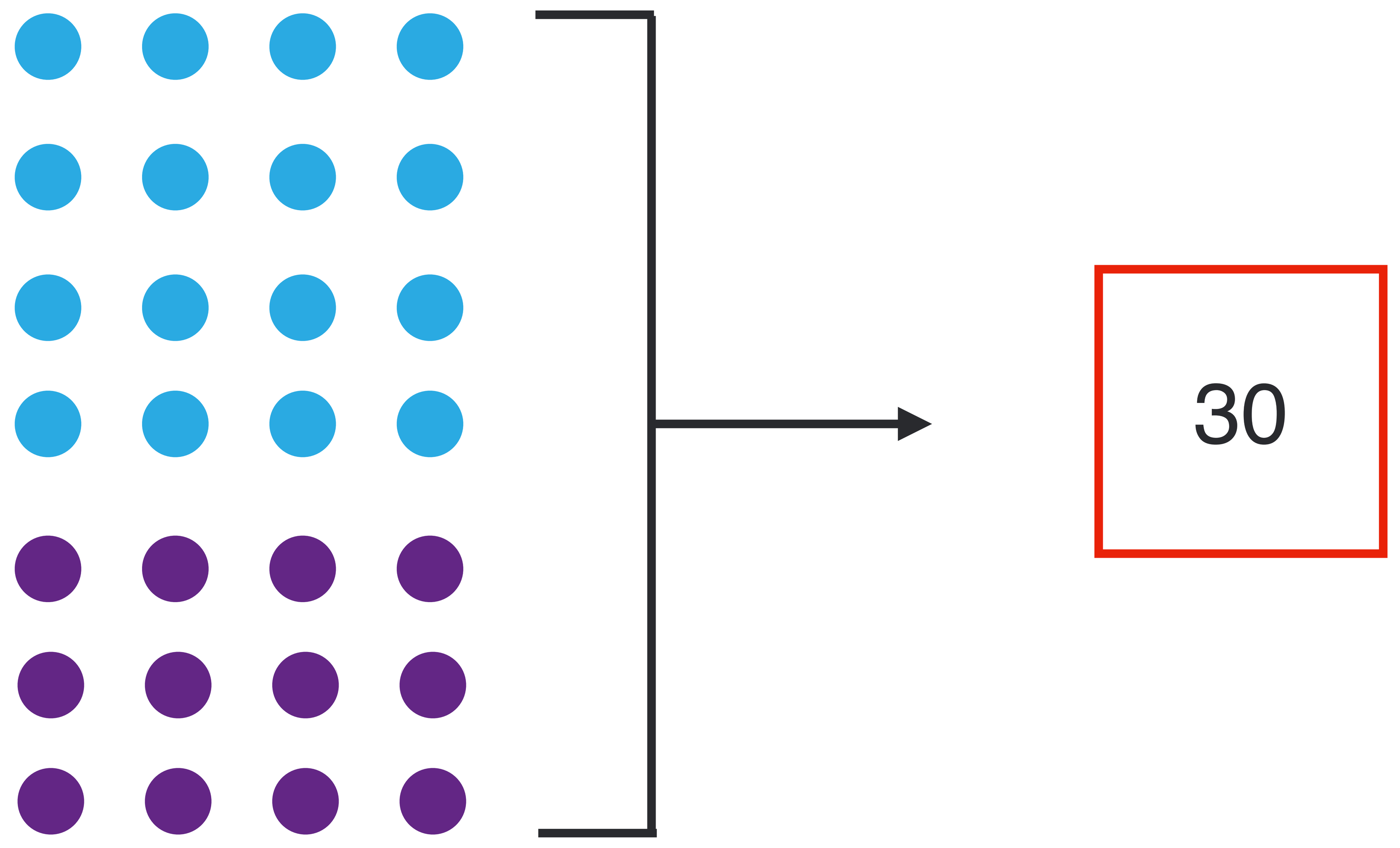



The following properties must hold even if up to $f < n/3$ nodes misbehave:

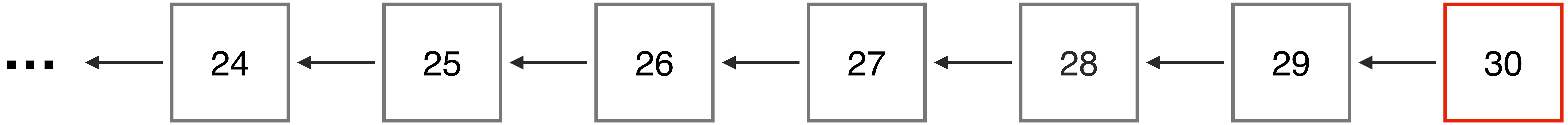
- **Agreement (Safety)**: For any *i*, If two (honest) replicas think that the *i*th block is agreed upon, they must have the same block
- **Termination (Liveness)**: For any *i*, at some point every (honest) replica will think that the *i*th block is agreed upon
- **Validity**: all agreed upon blocks are valid

# Block Maker

● Message (user → canister) ● Message (canister → canister)



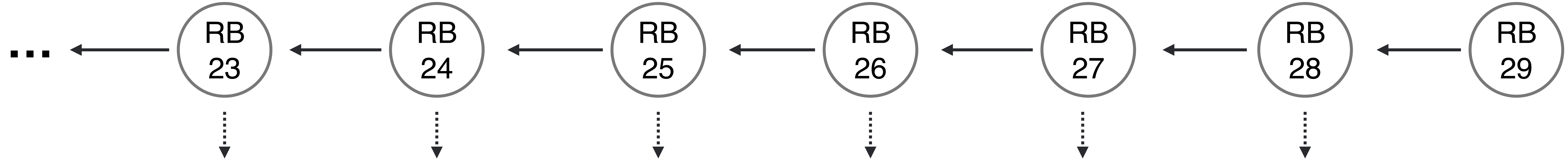

A block maker selects available messages and combines them into a block and broadcasts it




We need more than one block maker in each round, otherwise the IC would not be fault tolerant!

# Block Maker Ranking

A sequence of unpredictable random bits (called the random beacon) is used to rank block makers:




High Priority

| | Round 24 | Round 25 | Round 26 | Round 27 | Round 28 | Round 29 |
|---|---|---|---|---|---|---|
| **Rank 0** | Replica 1 | Replica 4 | Replica 2 | Replica 3 | Replica 4 | Replica 2 |
| **Rank 1** | Replica 4 | Replica 3 | Replica 3 | Replica 1 | Replica 2 | Replica 1 |
| **Rank 2** | Replica 3 | Replica 1 | Replica 4 | Replica 4 | Replica 1 | Replica 3 |
| **Rank 3** | Replica 2 | Replica 2 | Replica 1 | Replica 2 | Replica 3 | Replica 4 |

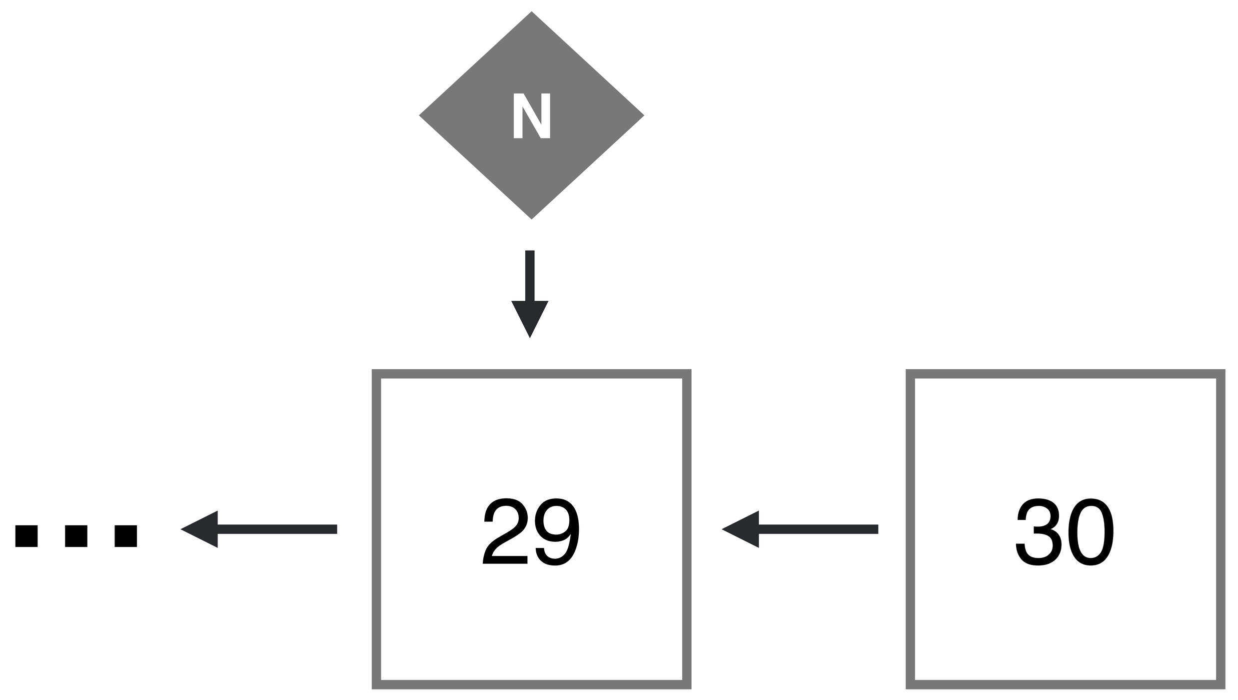
Low Priority

# Notarization

The notarization process ensures that a *valid* block proposal is published for every round

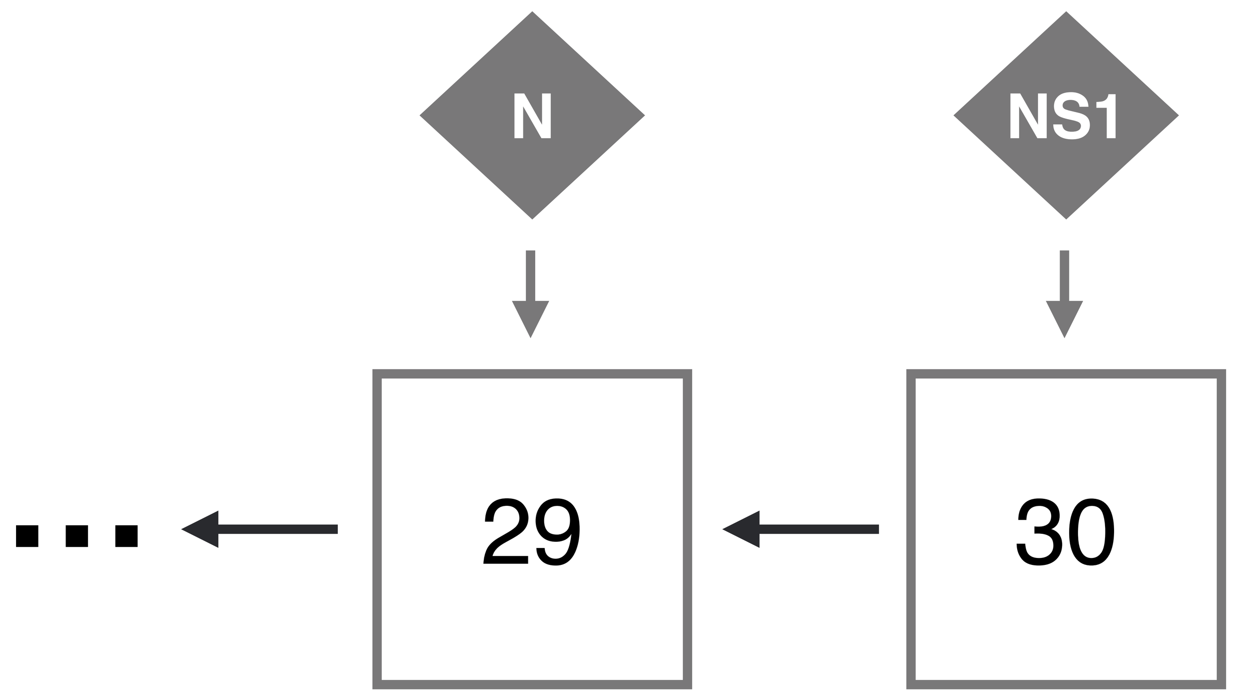
### Step 1

Replica 1 receives a block proposal for height 30, building on some notarized height 29 block



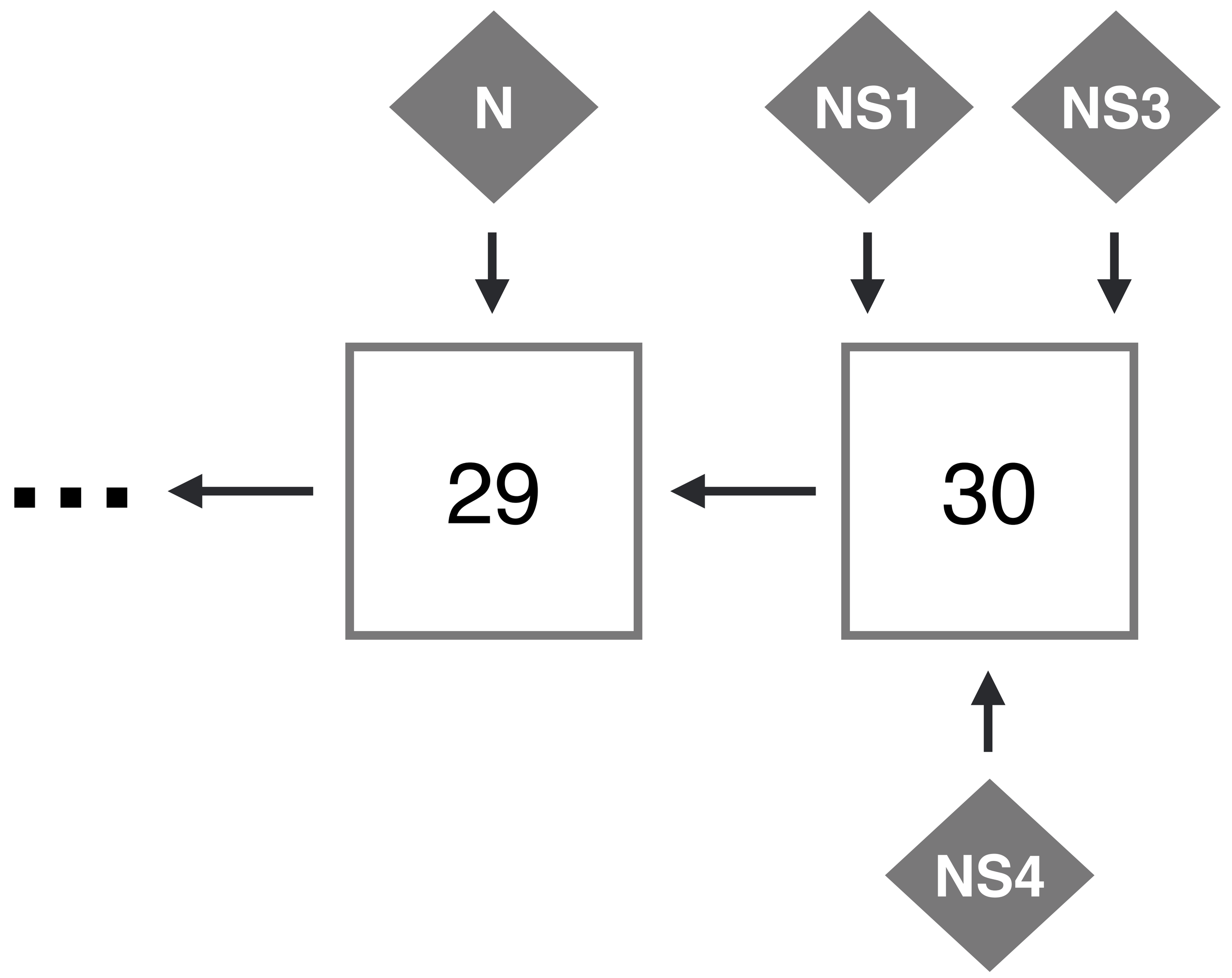

### Step 2

Replica 1 sees that the block is valid, signs it, and broadcasts its *notarization* share



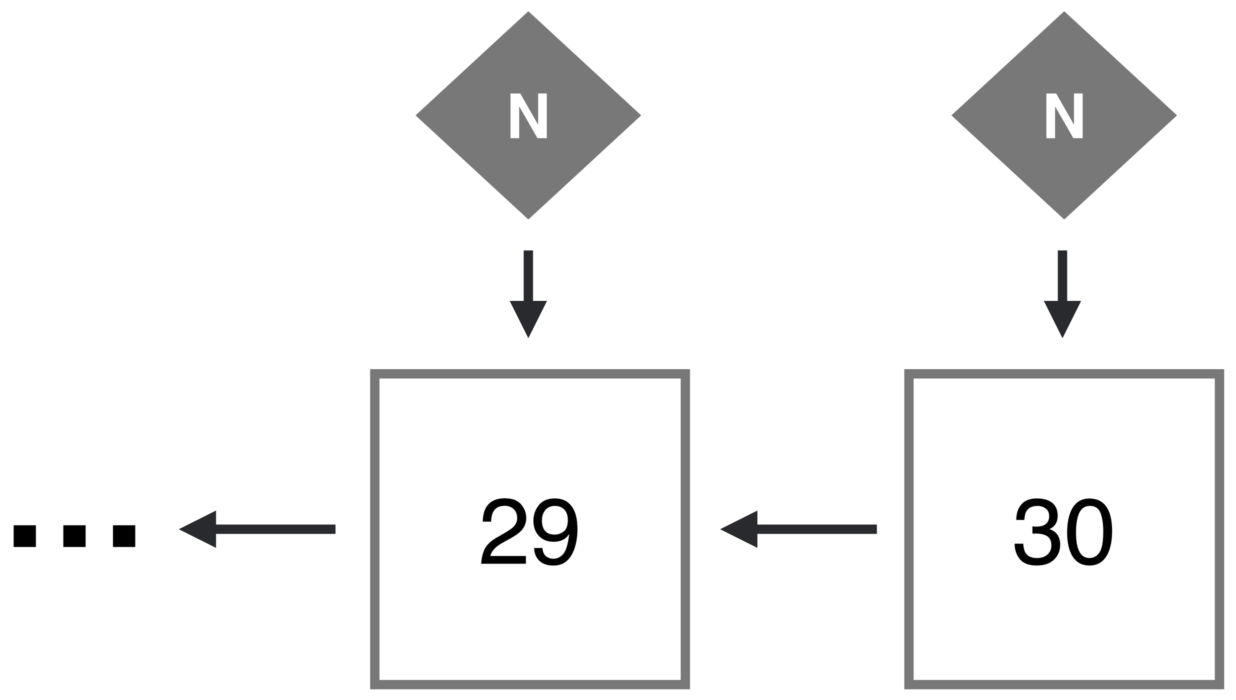

### Step 3

Replica 1 sees that replicas 3 and 4 also published their notarization shares on the block




### Step 4

3 notarization shares are sufficient approval: the shares are aggregated into a single full notarization. Block 30 is now notarized, and notaries wait for height 31 blocks

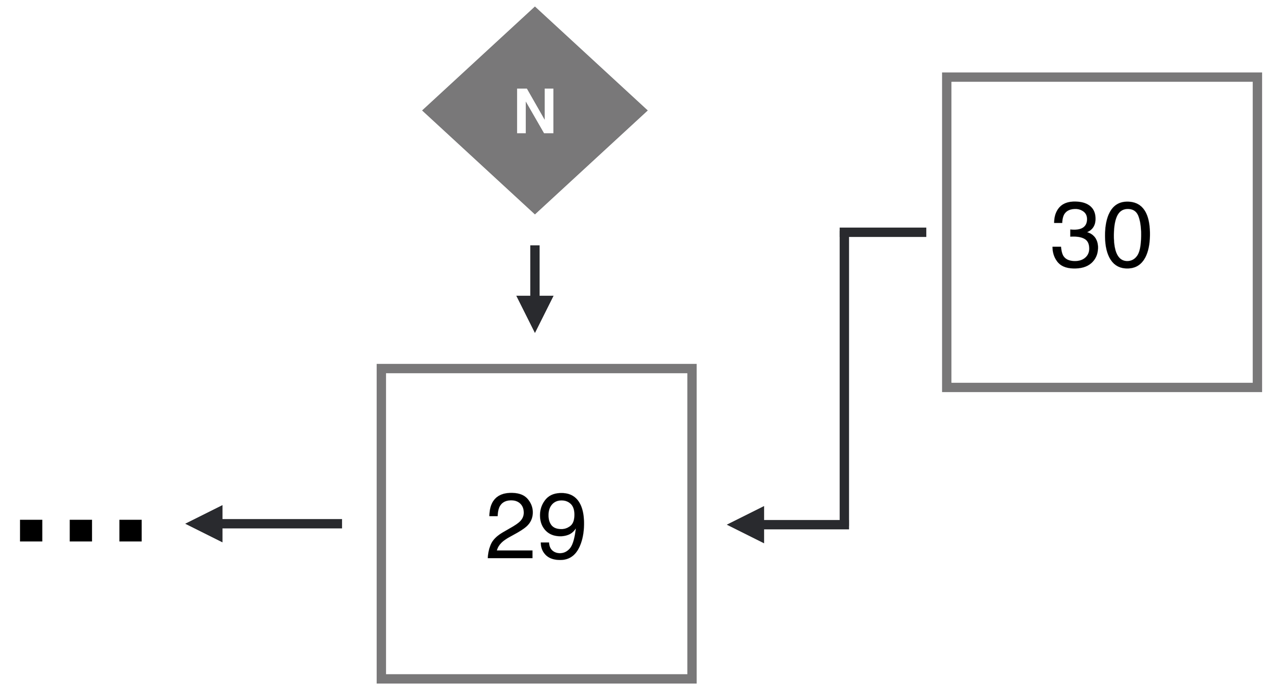


# Notarization

Replicas may notary-sign multiple blocks to ensure that at least one block becomes fully notarized

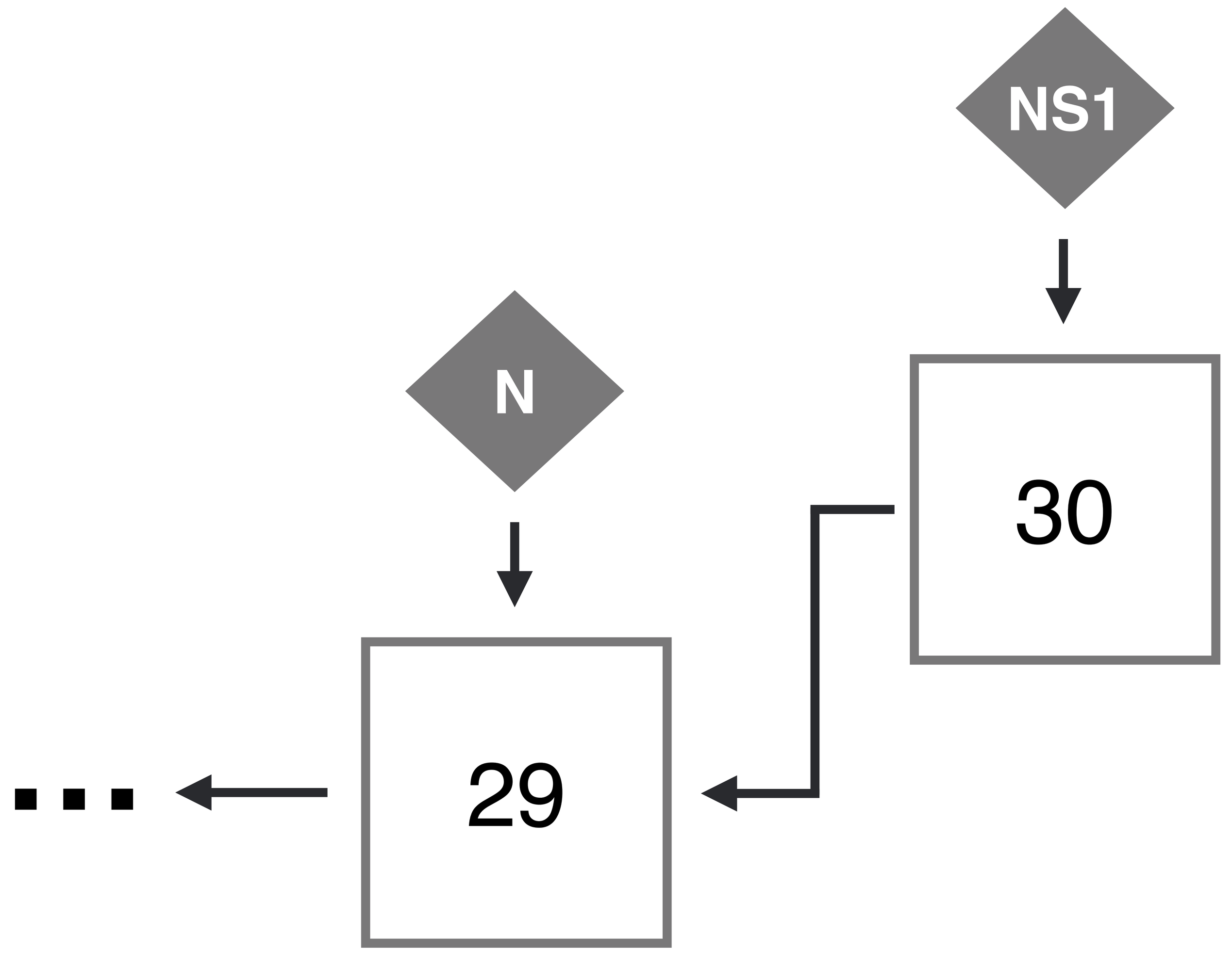
**Step 1**

Replica 1 receives a block proposal for height 30, building on some notarized height 29 block



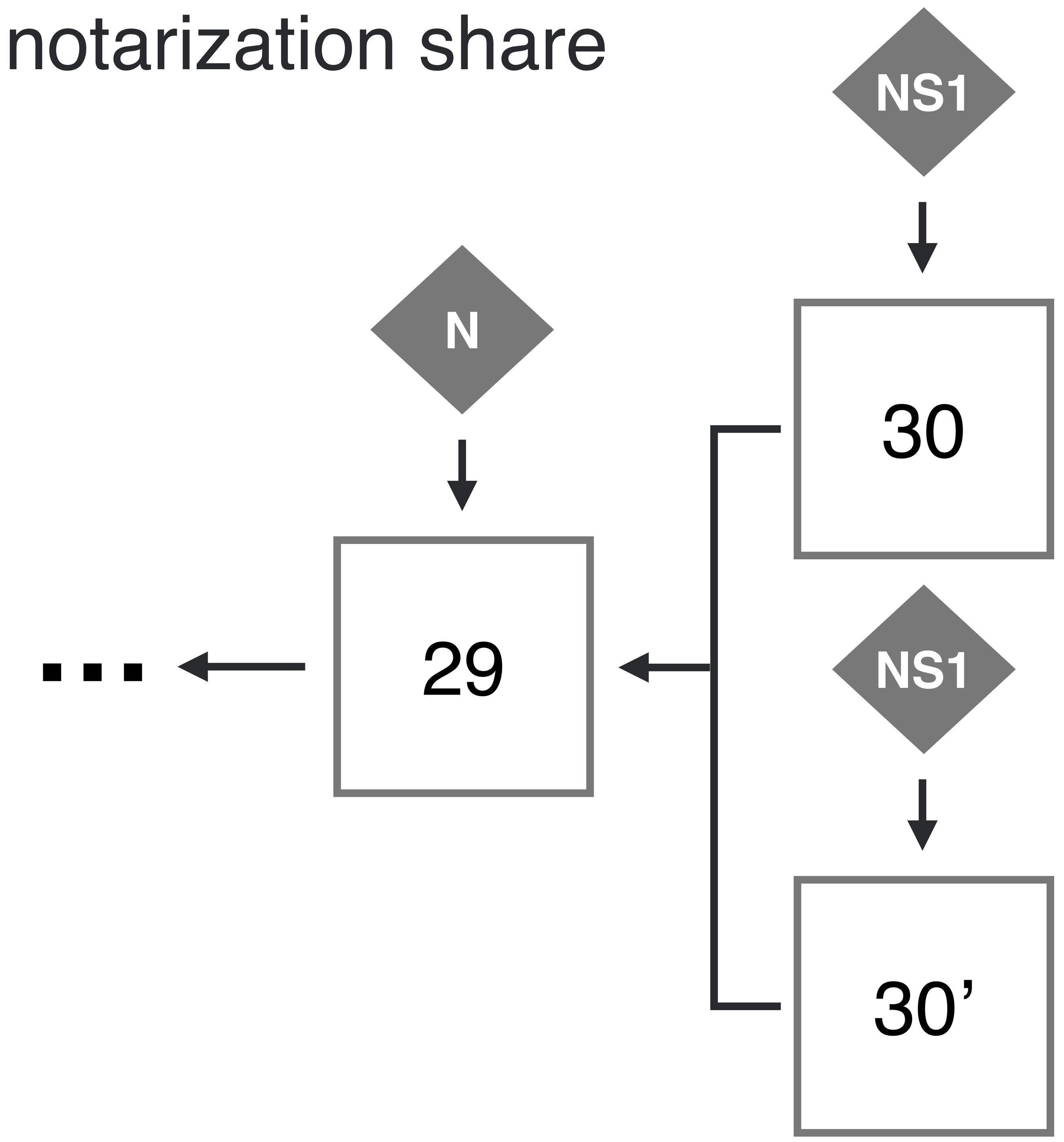

**Step 2**

Replica 1 sees that the block is valid, signs it, and broadcasts its *notarization* share



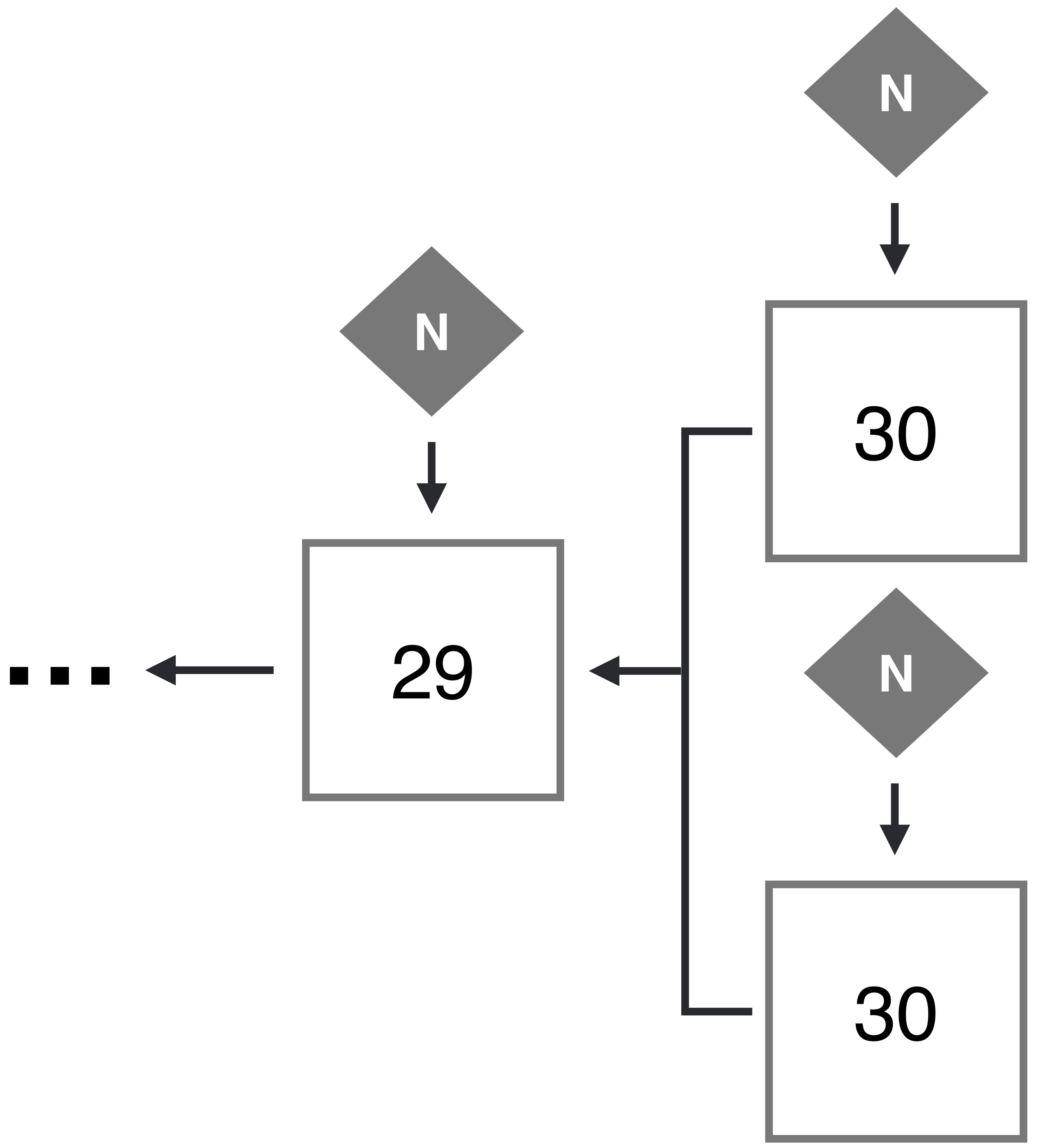

**Step 3**

Replicas 1 sees another height 30 block, which is also valid, and it broadcasts another notarization share



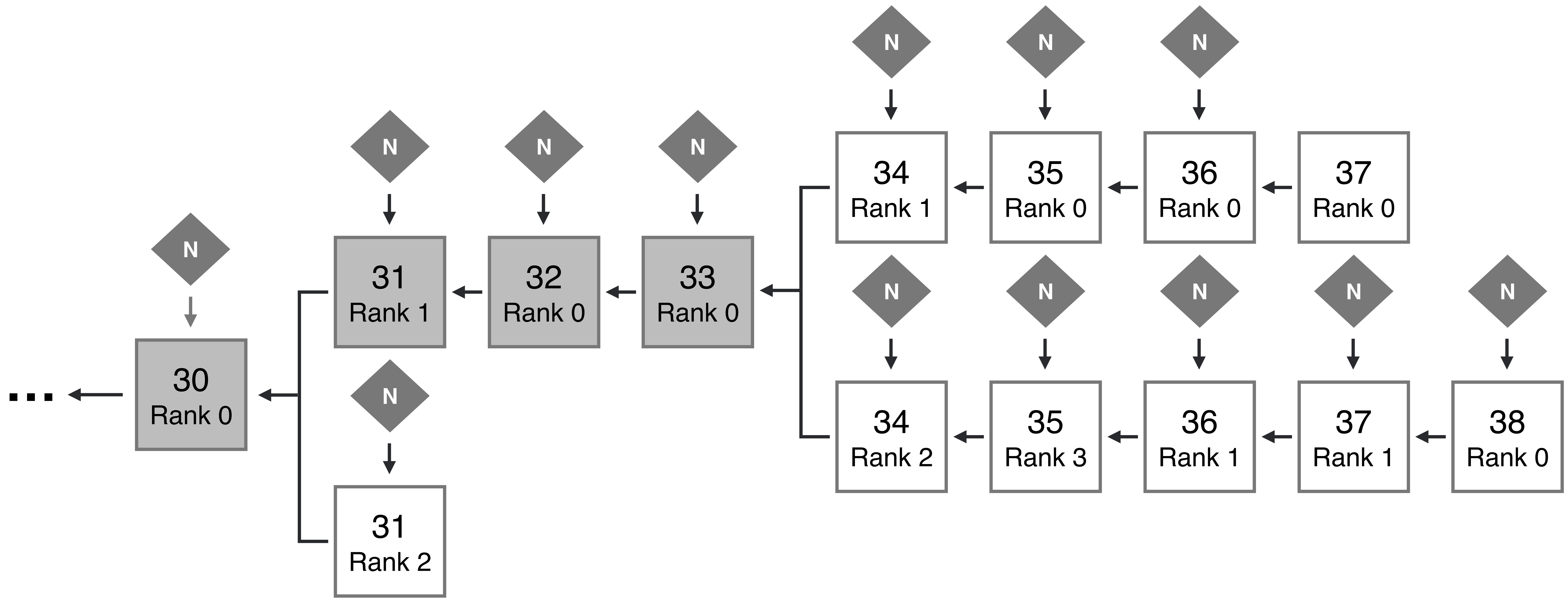

**Step 4**

Both height 30 blocks get enough support to become notarized



# Notarization with Block Maker Ranking

Multiple notarized blocks may exist at the same height!




One notarized block *b* at a height *h* = Agreement up to *h*

# Finalization

Replicas create finalization shares if they did not sign any other block at that height

**Step 1**

Replica 1 notary-signs block *b* at height 30

**Step 2**

Replica 1 observes that block *b* is fully notarized and will no longer notary-sign blocks at height ≤ 30

**Step 3**

Since replica 1 did not notary-sign any other block than block b, it signs block *b*, creating a finalization-share on *b*

**Step 4**

Replicas 2 and 4 also cast finalization shares on block *b*

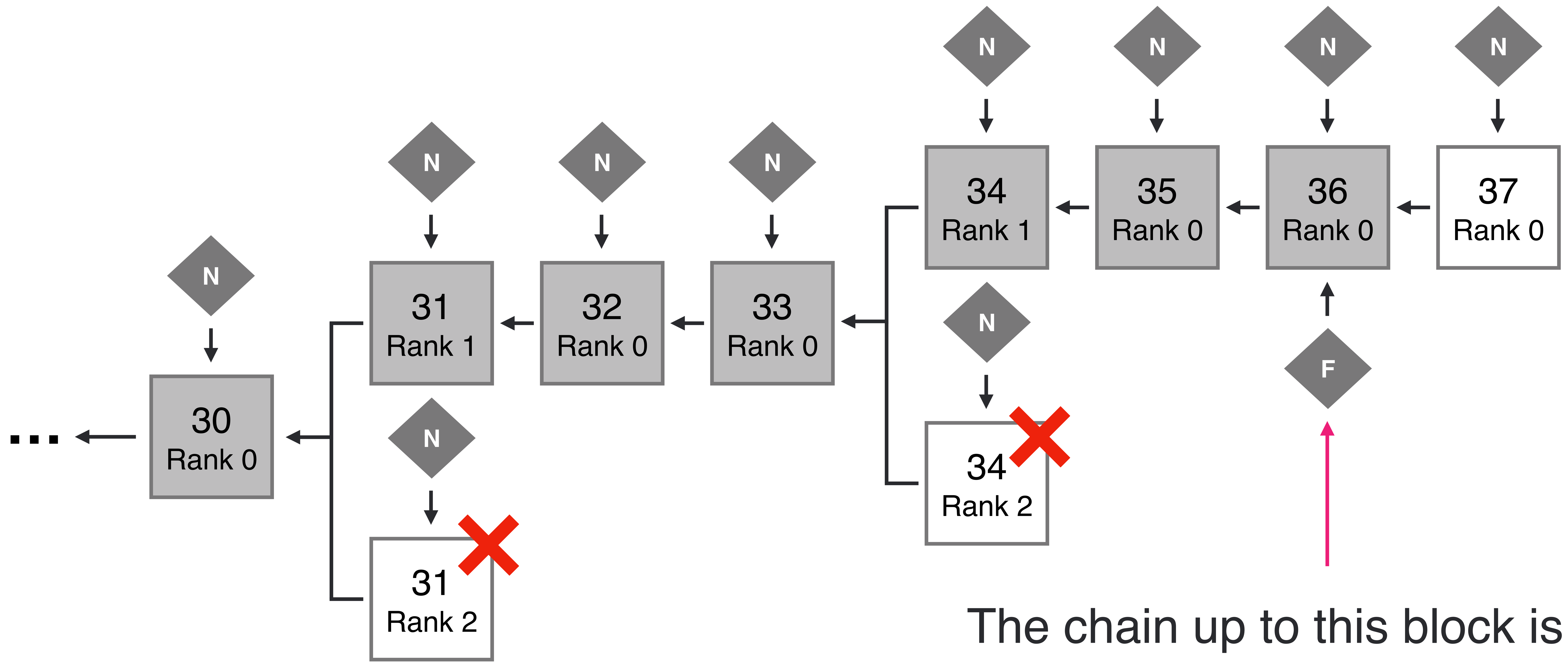
**Step 5**

3 finalization-shares are sufficient approval: the shares are aggregated into a single full finalization


Replica 1 did not notary-sign any height 30 block other than b

# Finalization

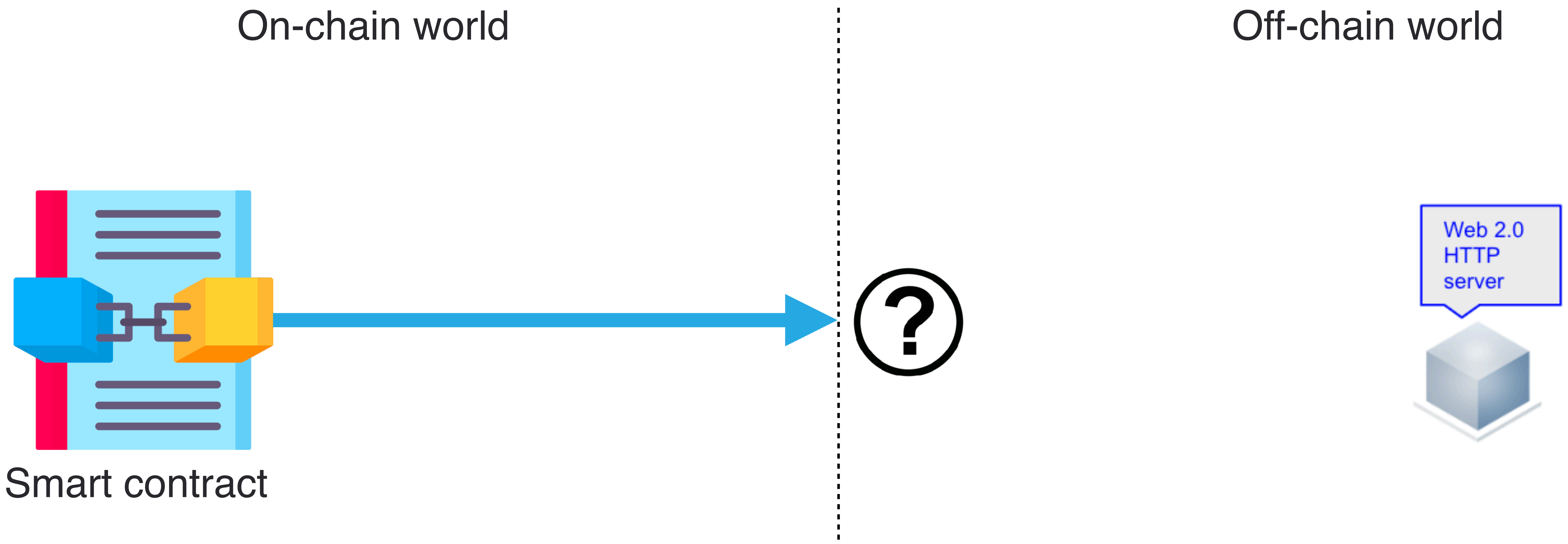
Finalization on block *b* at height *h* = Proof that no other block is notarized at height *h*




The chain up to this block is final

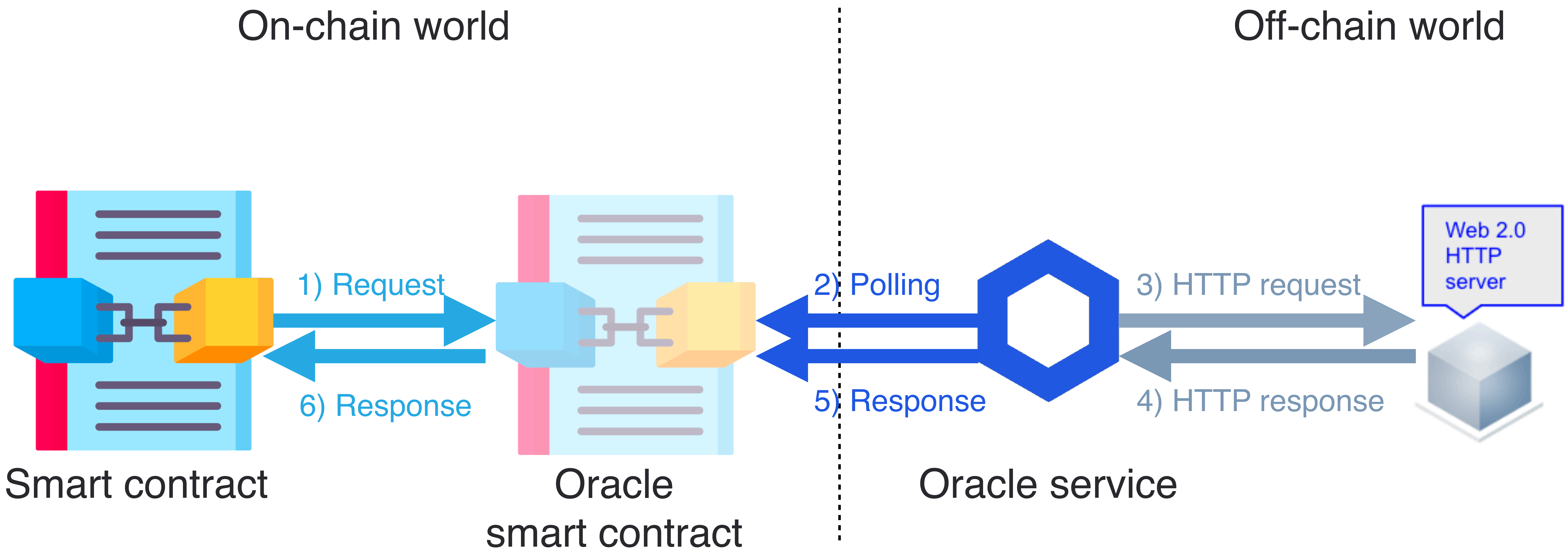
# Closer Look: HTTPS Outcalls

# Interaction with Web 2.0




**Smart contracts cannot access Web 2.0 resources directly!**

# Interaction with Web 2.0




An oracle infrastructure is required.
**The oracles must be trusted!**

# Interaction with Web 2.0 on the Internet Computer

On-chain world

Off-chain world


- Replicas can communicate with web servers directly (no intermediary!)
- Responses go through consensus to ensure deterministic behavior

HTTPS requests & responses sent and received by replicas

# The Internet Computer Today

# Live Since May 2021!

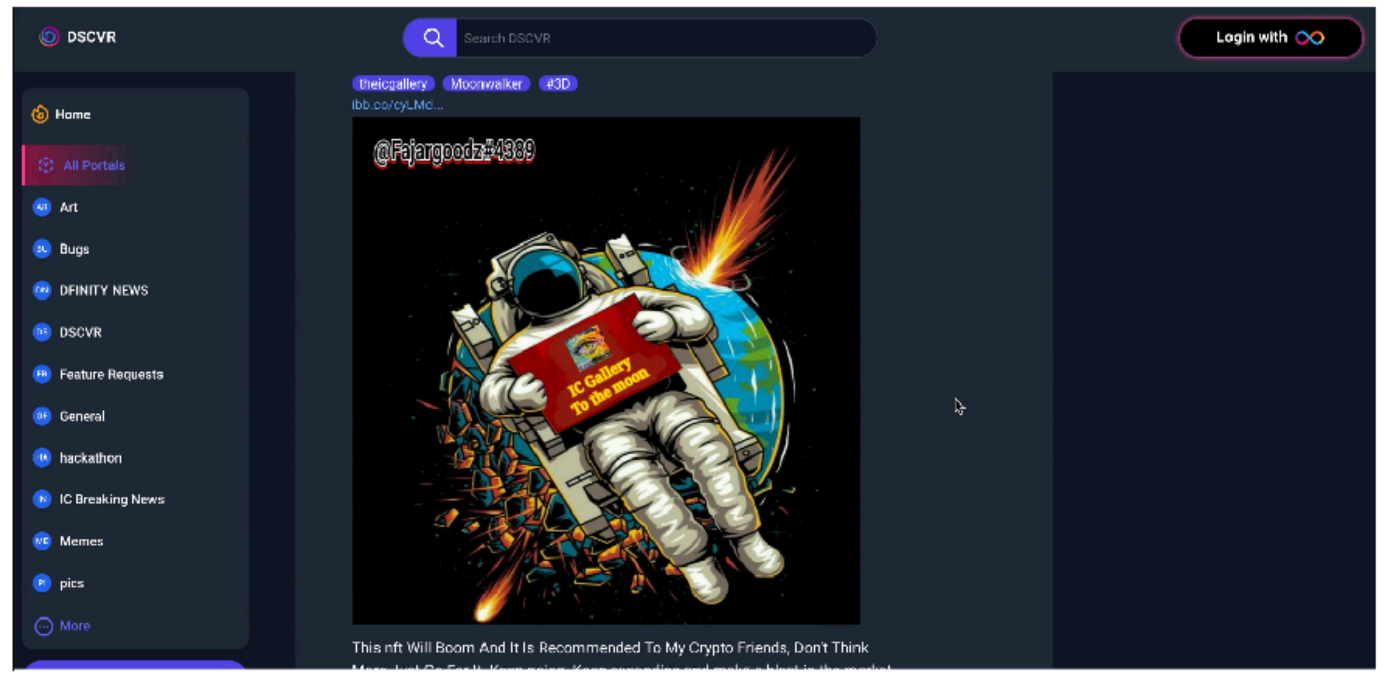

| | |
|---|---|
| Total Canister State | 2.57 TB |
| Node Providers | 81 |
| Subnets | 36 |
| Boundary Nodes | 16 |
| Total Nodes | 1’236 |
| Nodes in Subnets | 549 |
| Internet Identity Anchors | 2’094’038 |

| | |
|---|---|
| Total Supply | 492’585’902 ICP |
| Circulating Supply | 274’588’059 ICP |
| ICP Transactions | 5’046’886 |
| Burn ICP Transactions | 96’756 ICP |
| Burn Fees | 444 ICP |
| NNS Proposals | 96’096 |

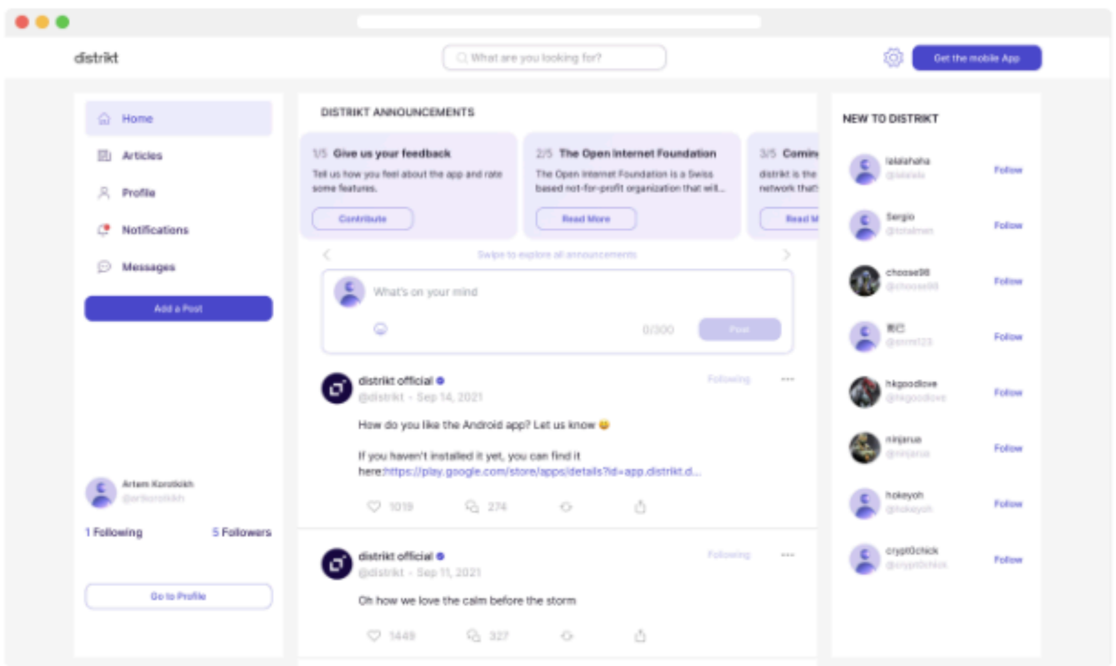
https://dashboard.internetcomputer.org

# Growing Blockchain Ecosystem

## DSCVR
dscvr.one

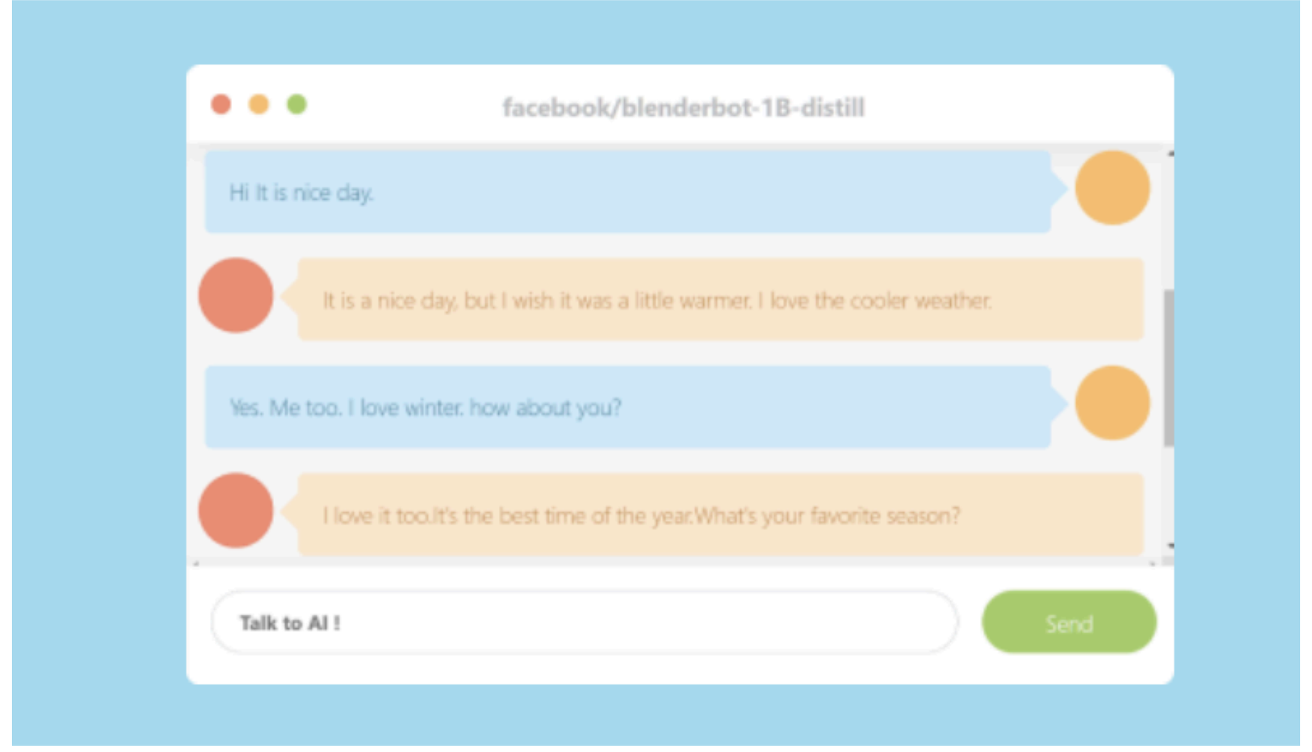


## Distrikt
az5sd-cqaaa-aaaae-aaarq-cai

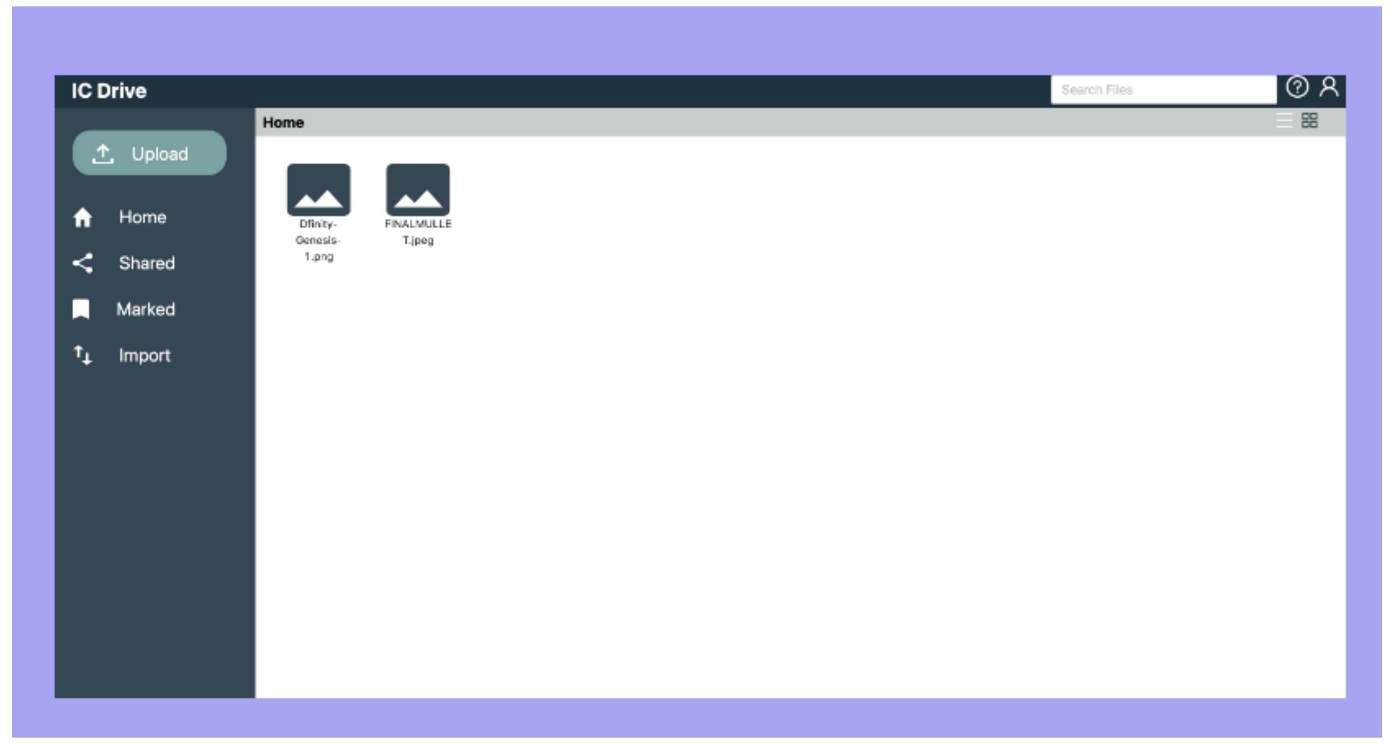


## Internet Identity
identity.ic0.app

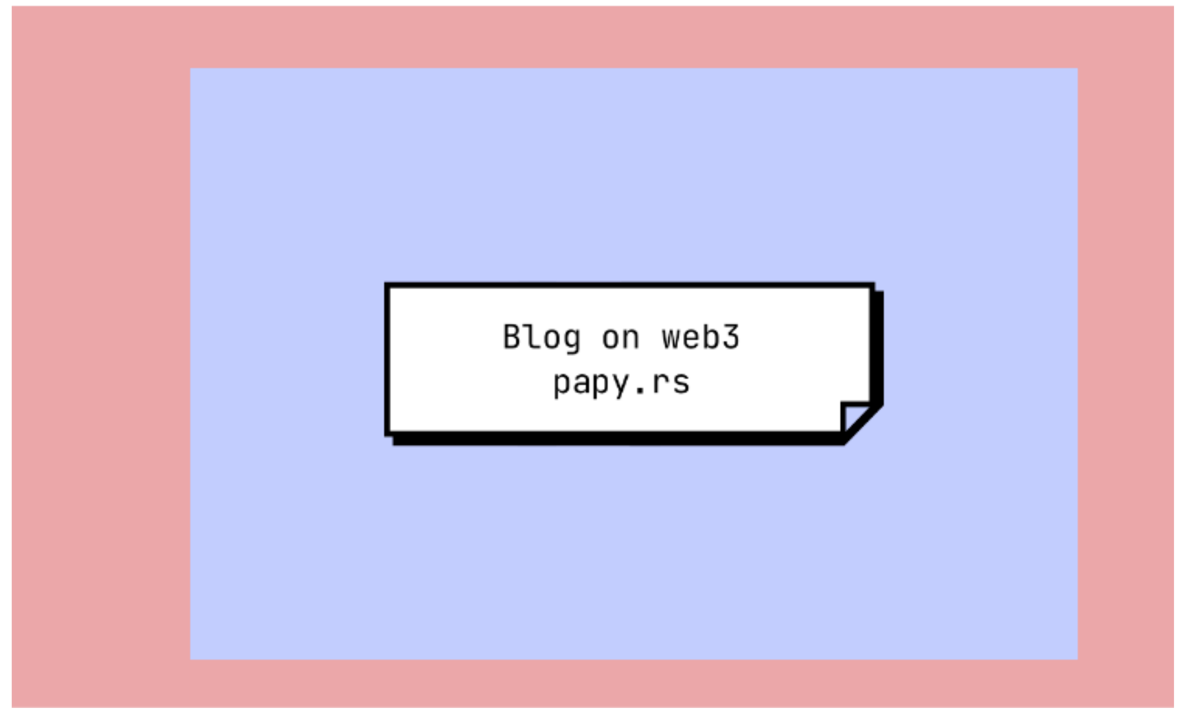
## OpenChat
oc.app



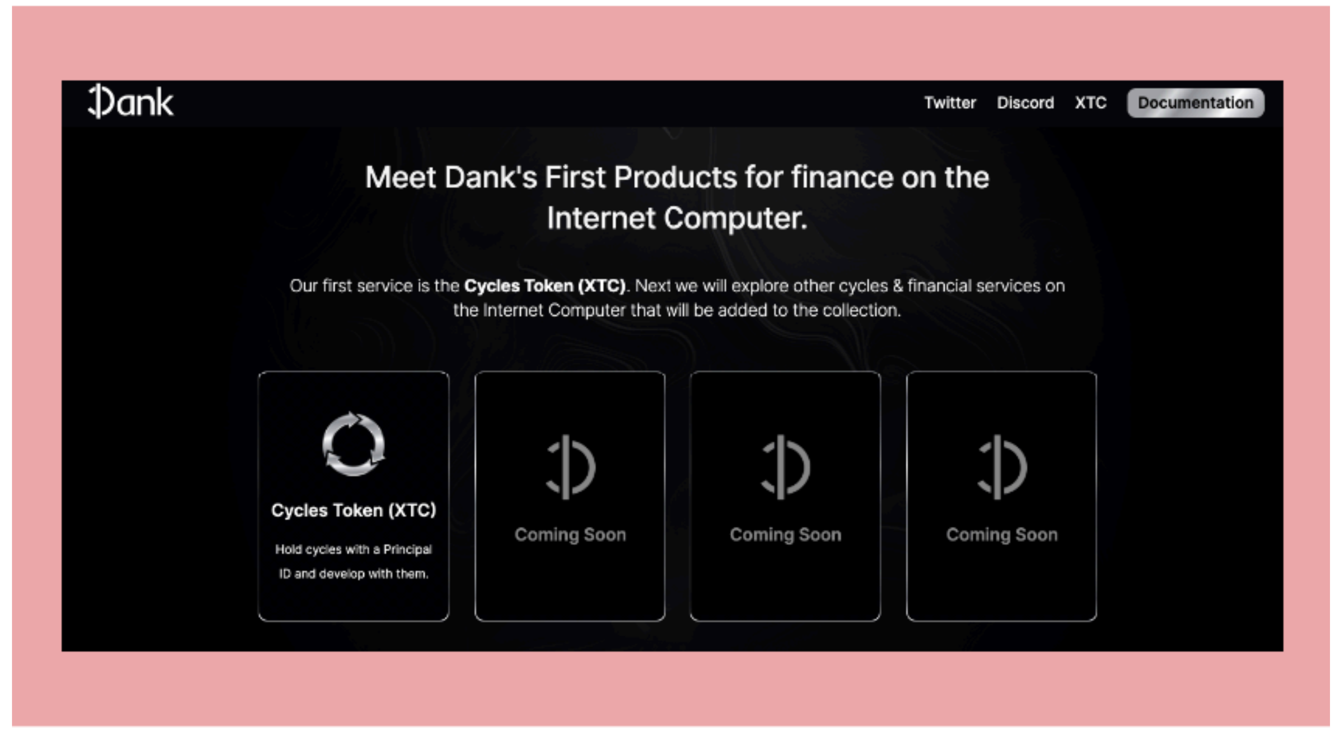

## IC Drive
icdrive.co



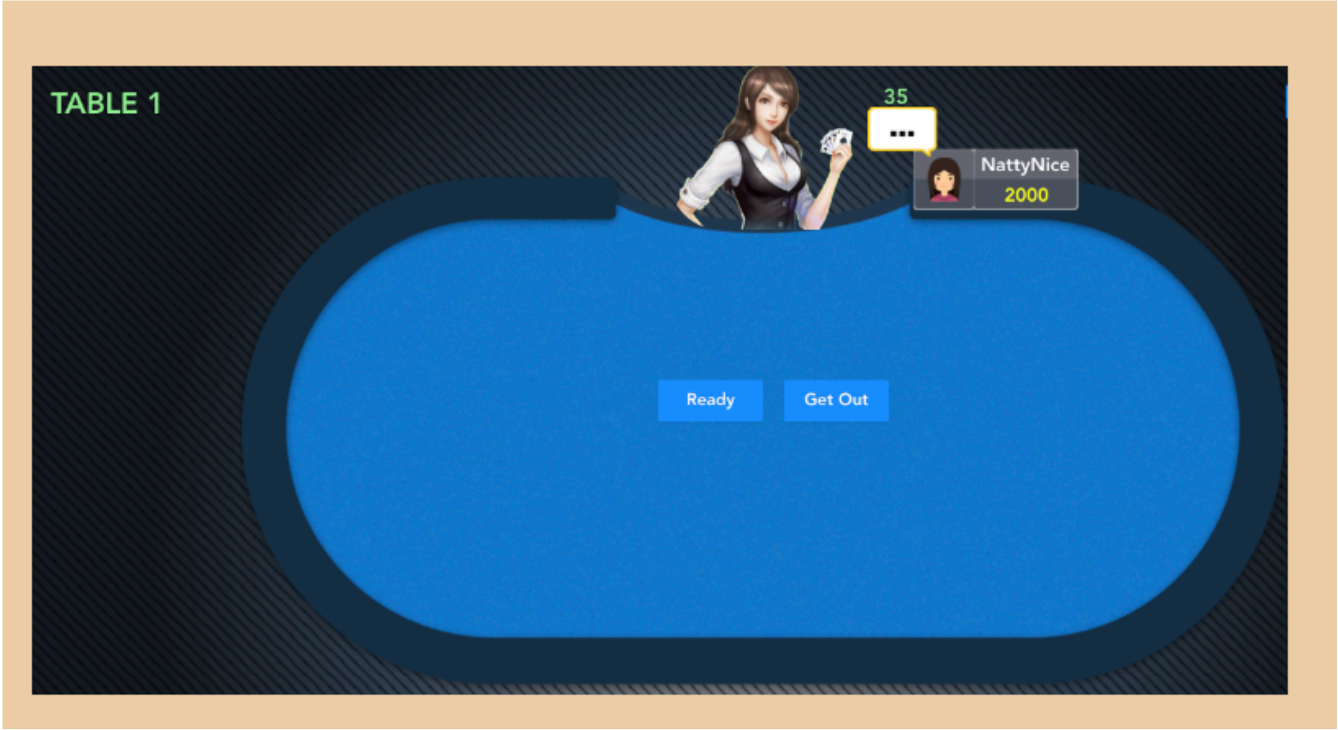

## Papyrs
app.papy.rs




## Dank
dank.ooo




## Texas Hold'em
lm5fh-ayaaa-aaaah-aafua-cai




https://internetcomputer.org/showcase

# Many Distributed Systems Problems

- Disseminating messages among nodes in the same subset
- Disseminating messages between subnets
- Scheduling and concurrent execution of canister messages
- Enabling nodes to catch up
- Handling churn (adding/removing nodes)
- Guaranteeing consistency
- Upgrading to next protocol version
- Creating new subnets
- Load balancing
- …

# Internet Computer vs. …

| | | | |
|---|---|---|---|
| Average block time | 1 block / 10 minutes | 1 block / 15 seconds | 37 blocks / second |
| | | | |
| Finality | 1 hour | 3 minutes | 1-3 seconds |
| | | | |
| TXs per second | 7 | 15 | 30,000 (write)<br>2,000,000 (read) |
| | | | |
| Validation data | 440 GB | 750 GB | 48 bytes |

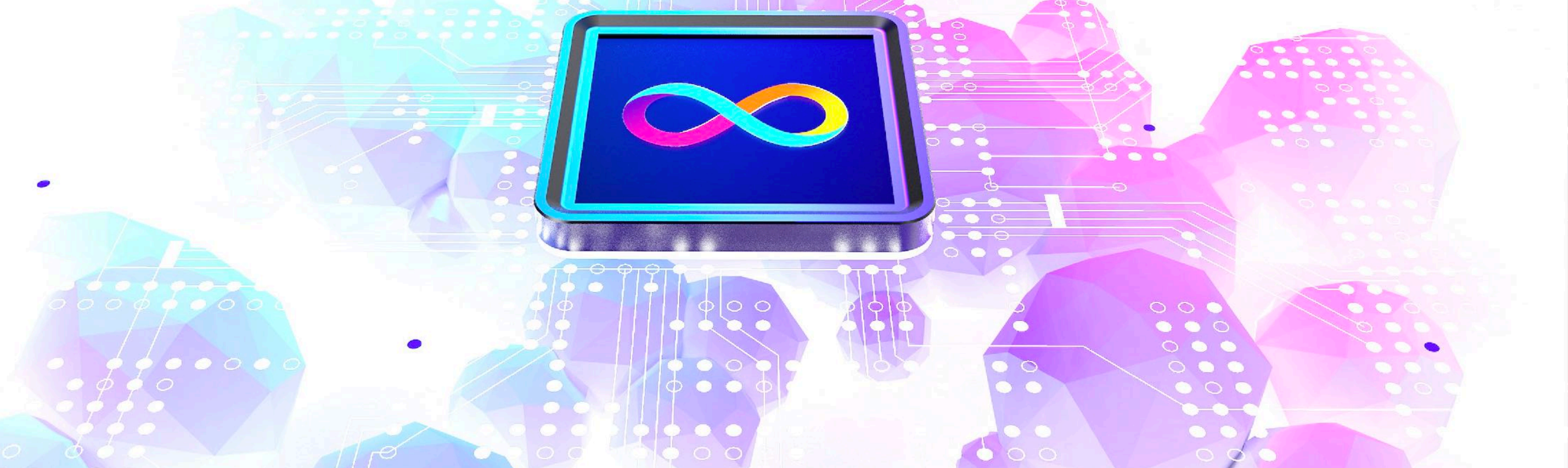
# More information

- Infographic: https://internetcomputer.org/icig.pdf

- Technical Library:
  - https://www.youtube.com/playlist?list=PLuhDt1vhGcrfHG_rnRKsqZO1jL_Pd970h (videos)
  - https://medium.com/dfinity (blogposts)

- 200,000,000 CHF Developer Grant Program: https://dfinity.org/grants/

- DFINITY SDK: https://internetcomputer.org/docs/current/developer-docs/build/



DFINITY