# Consensus on Demand

Jakub Sliwinski, Yann Vonlanthen, Roger Wattenhofer

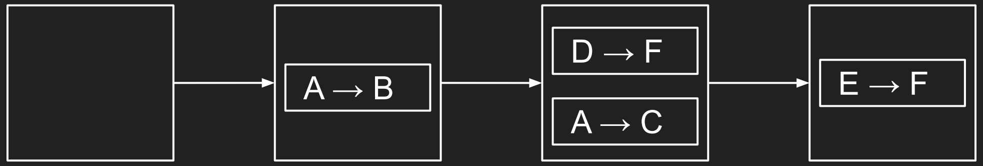
*ETH Zurich - **Dis**tributed **Co**mputing Group*

# Bitcoin: A Peer-to-Peer Electronic Cash System

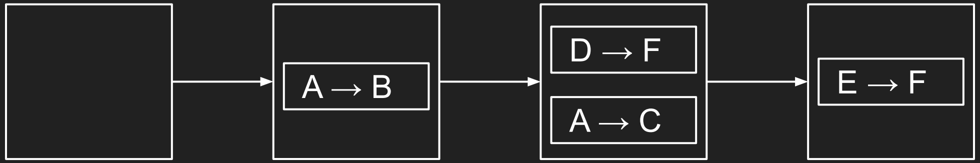
Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

**Abstract.** A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

# Why do we need a blockchain?

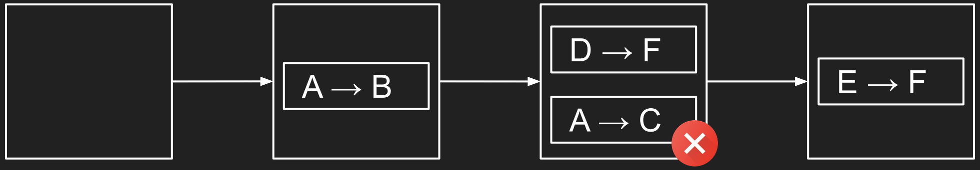


# Why do we need a blockchain?



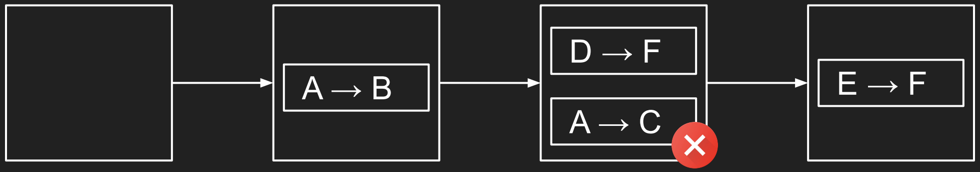

Prevent Double-Spending

# Why do we need a blockchain?




Prevent Double-Spending

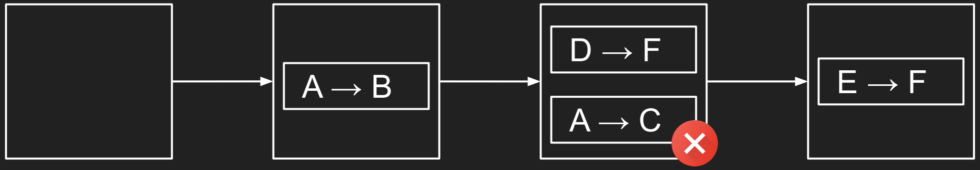
# Why do we need a blockchain?




Prevent Double-Spending

Total Order

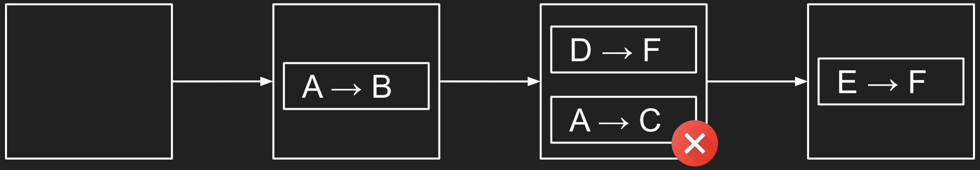
# Why do we need a blockchain?




Prevent Double-Spending

Total Order = Consensus

# Why do we need a blockchain?

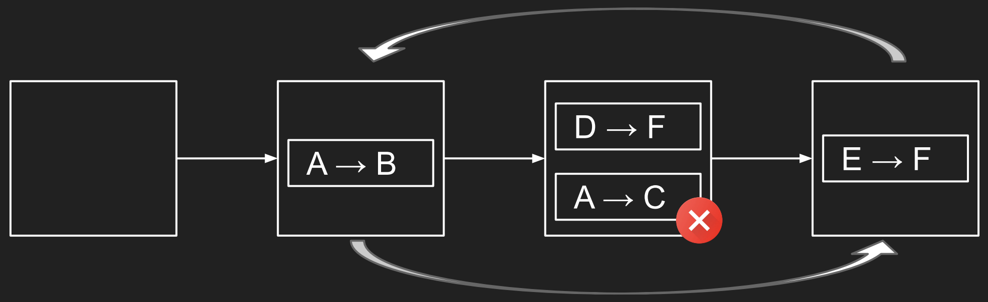



Prevent Double-Spending

$\neq$

Total Order = Consensus

# Why do we need a blockchain?



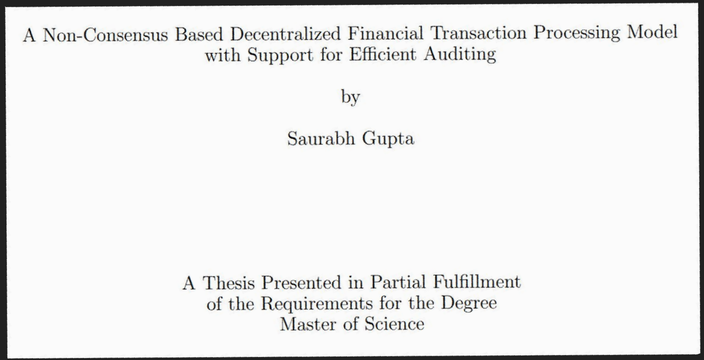

Prevent Double-Spending

$\neq$

Total Order = Consensus

# Online Payments Without Consensus

# Online Payments Without Consensus



A Non-Consensus Based Decentralized Financial Transaction Processing Model with Support for Efficient Auditing

by

Saurabh Gupta

A Thesis Presented in Partial Fulfillment of the Requirements for the Degree Master of Science

# Online Payments Without Consensus

A Non-Consensus Based Decentralized Financial Transaction Processing Model with Support for Efficient Auditing

by

Saurabh Gupta

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

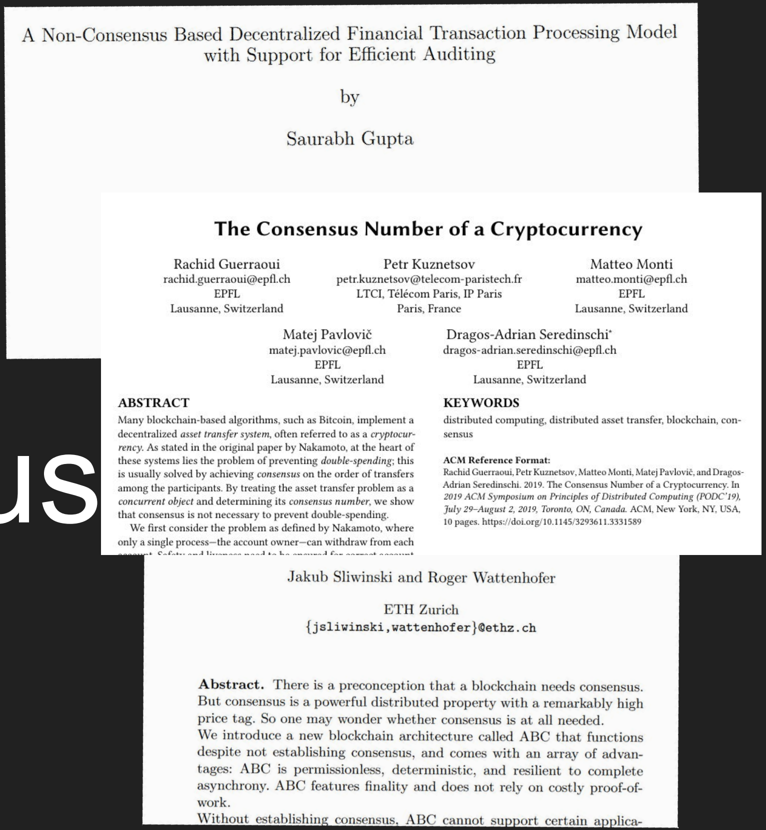
**ABC: Asynchronous Blockchain without Consensus**

Jakub Sliwinski and Roger Wattenhofer

ETH Zurich
{jsliwinski,wattenhofer}@ethz.ch

**Abstract.** There is a preconception that a blockchain needs consensus. But consensus is a powerful distributed property with a remarkably high price tag. So one may wonder whether consensus is at all needed.
We introduce a new blockchain architecture called ABC that functions despite not establishing consensus, and comes with an array of advantages: ABC is permissionless, deterministic, and resilient to complete asynchrony. ABC features finality and does not rely on costly proof-of-work.
Without establishing consensus, ABC cannot support certain applica-

# Online Payments Without Consensus



A Non-Consensus Based Decentralized Financial Transaction Processing Model with Support for Efficient Auditing

by

Saurabh Gupta

**The Consensus Number of a Cryptocurrency**

Rachid Guerraoui
rachid.guerraoui@epfl.ch
EPFL
Lausanne, Switzerland

Petr Kuznetsov
petr.kuznetsov@telecom-paristech.fr
LTCI, Télécom Paris, IP Paris
Paris, France

Matteo Monti
matteo.monti@epfl.ch
EPFL
Lausanne, Switzerland

Matej Pavlovič
matej.pavlovic@epfl.ch
EPFL
Lausanne, Switzerland

Dragos-Adrian Seredinschi*
dragos-adrian.seredinschi@epfl.ch
EPFL
Lausanne, Switzerland

**ABSTRACT**

Many blockchain-based algorithms, such as Bitcoin, implement a decentralized *asset transfer system*, often referred to as a *cryptocurrency*. As stated in the original paper by Nakamoto, at the heart of these systems lies the problem of preventing *double-spending*; this is usually solved by achieving *consensus* on the order of transfers among the participants. By treating the asset transfer problem as a *concurrent object* and determining its *consensus number*, we show that consensus is not necessary to prevent double-spending.

We first consider the problem as defined by Nakamoto, where only a single process—the account owner—can withdraw from each

**KEYWORDS**

distributed computing, distributed asset transfer, blockchain, consensus

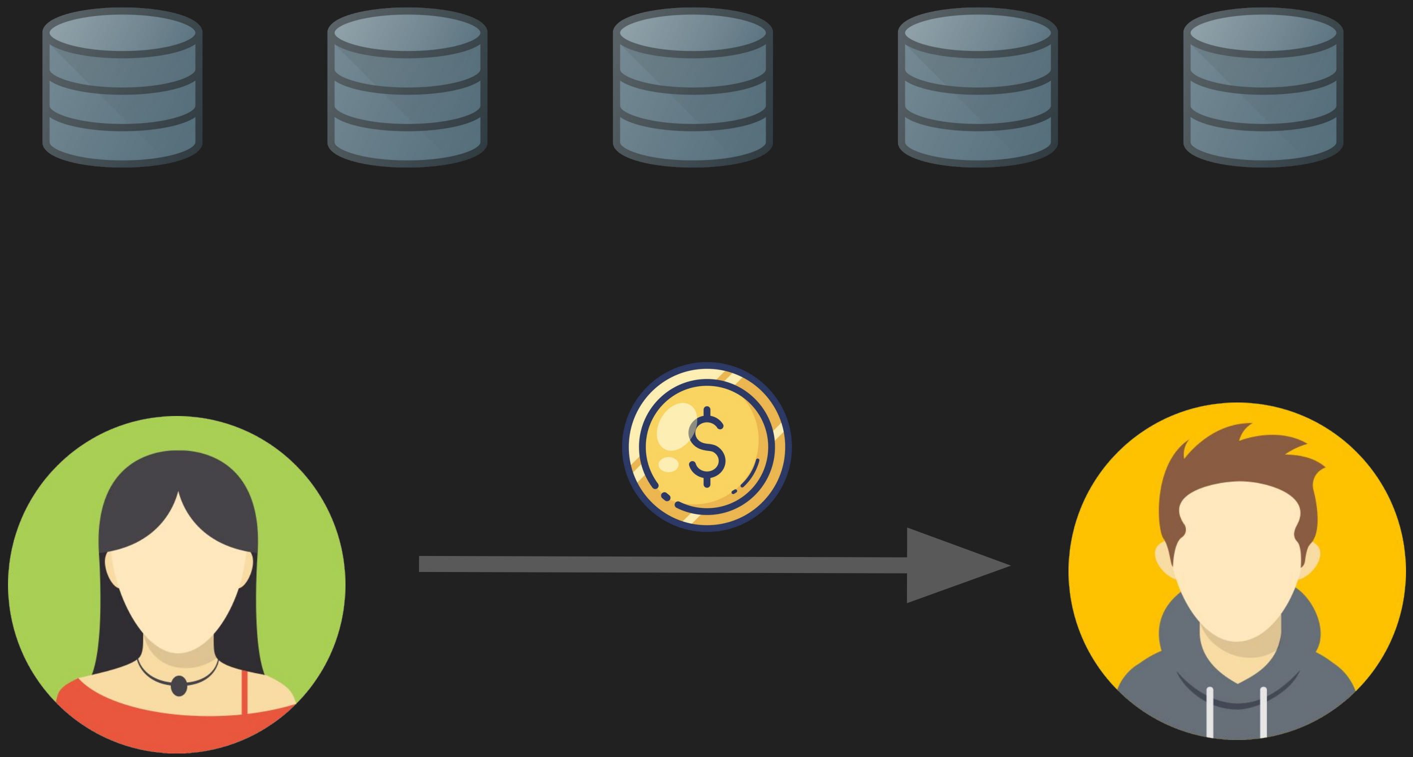
**ACM Reference Format:**

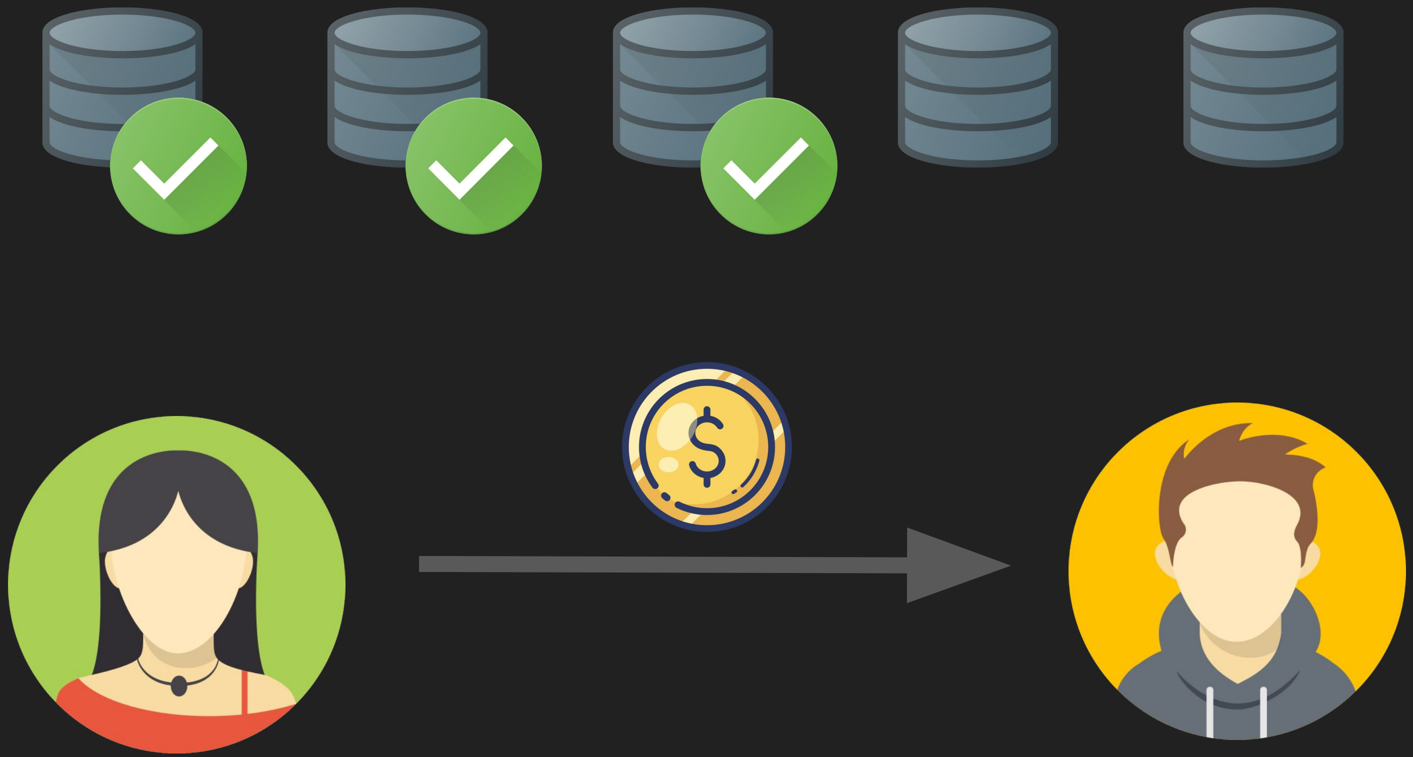
Rachid Guerraoui, Petr Kuznetsov, Matteo Monti, Matej Pavlovič, and Dragos-Adrian Seredinschi. 2019. The Consensus Number of a Cryptocurrency. In *2019 ACM Symposium on Principles of Distributed Computing (PODC'19), July 29–August 2, 2019, Toronto, ON, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3293611.3331589

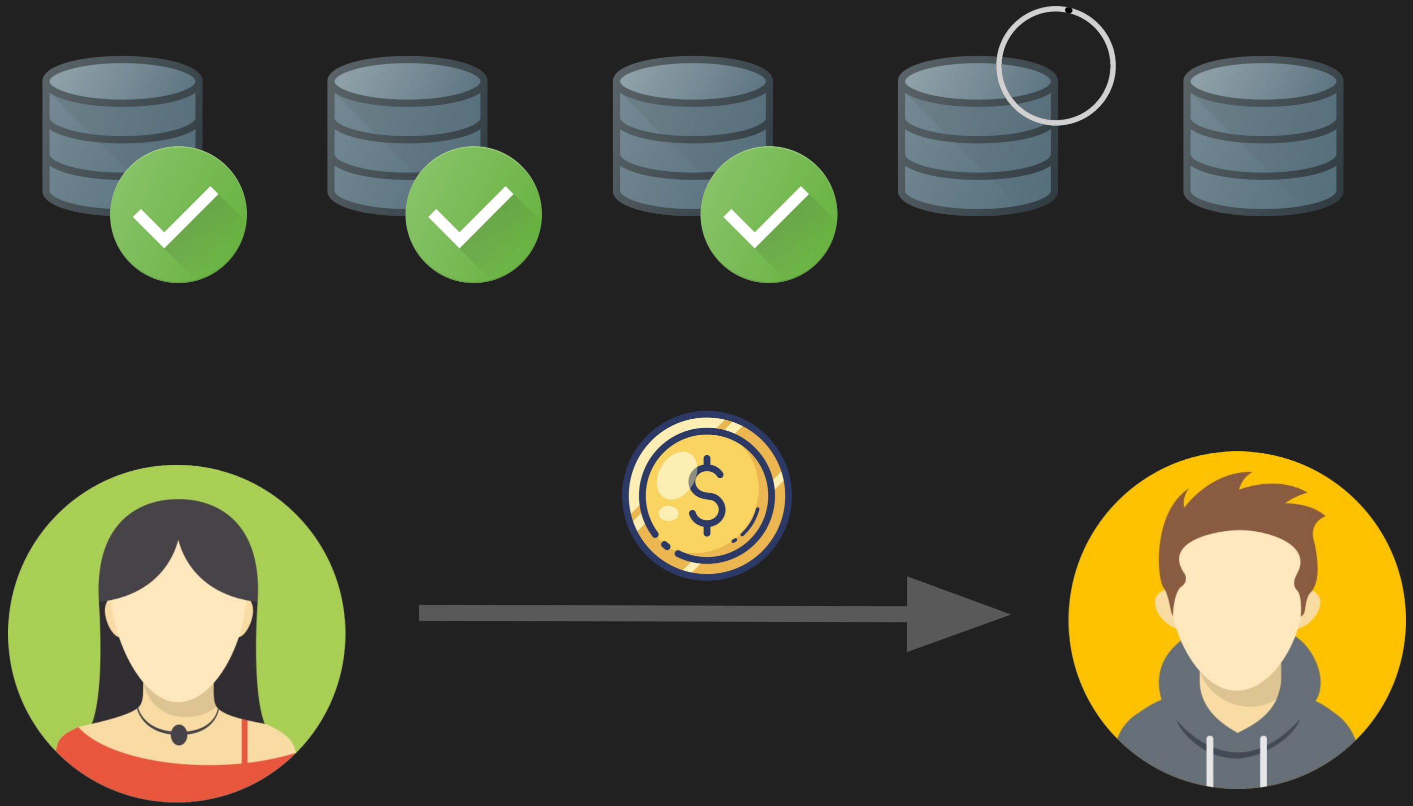
Jakub Sliwinski and Roger Wattenhofer

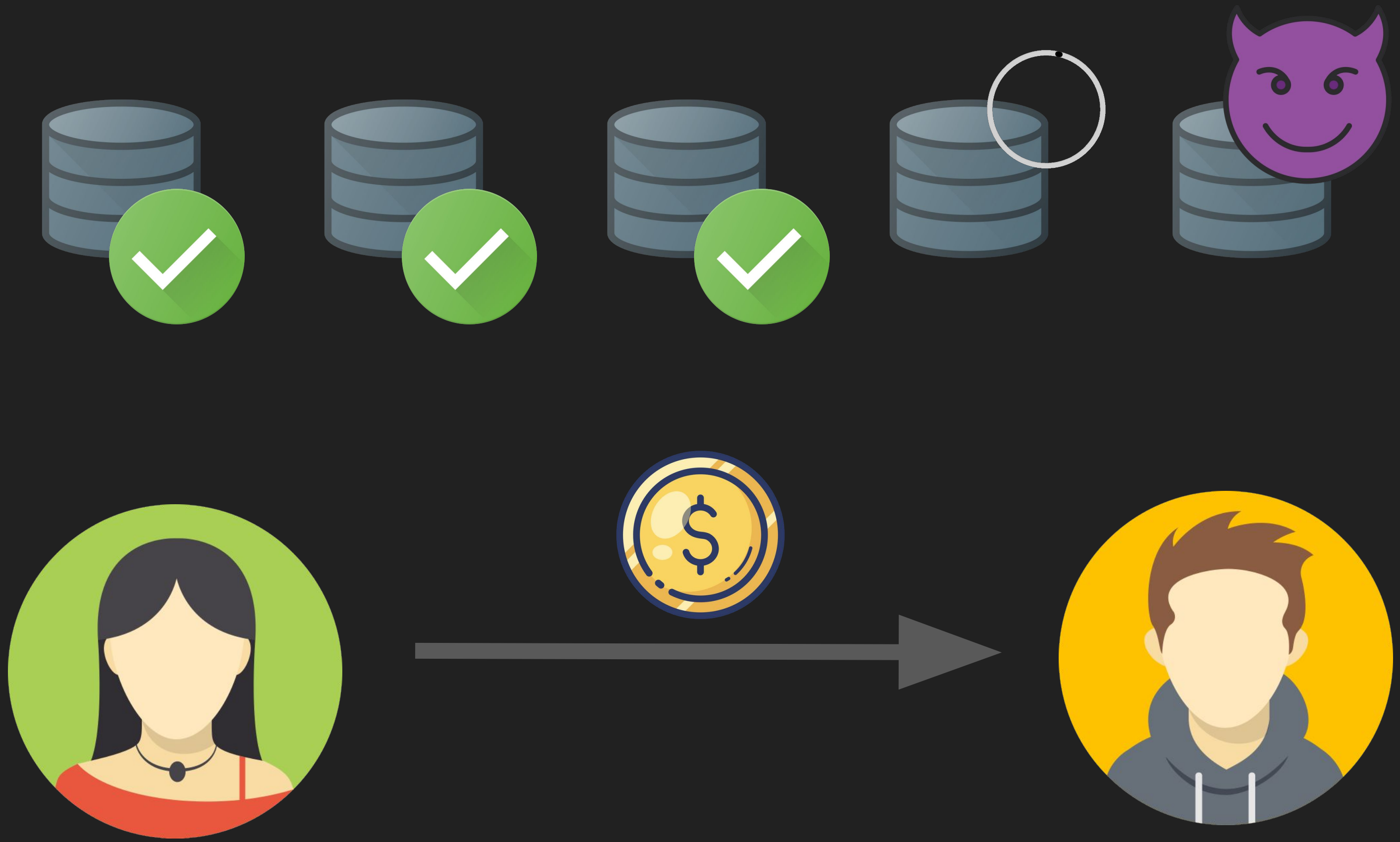
ETH Zurich
{jsliwinski,wattenhofer}@ethz.ch

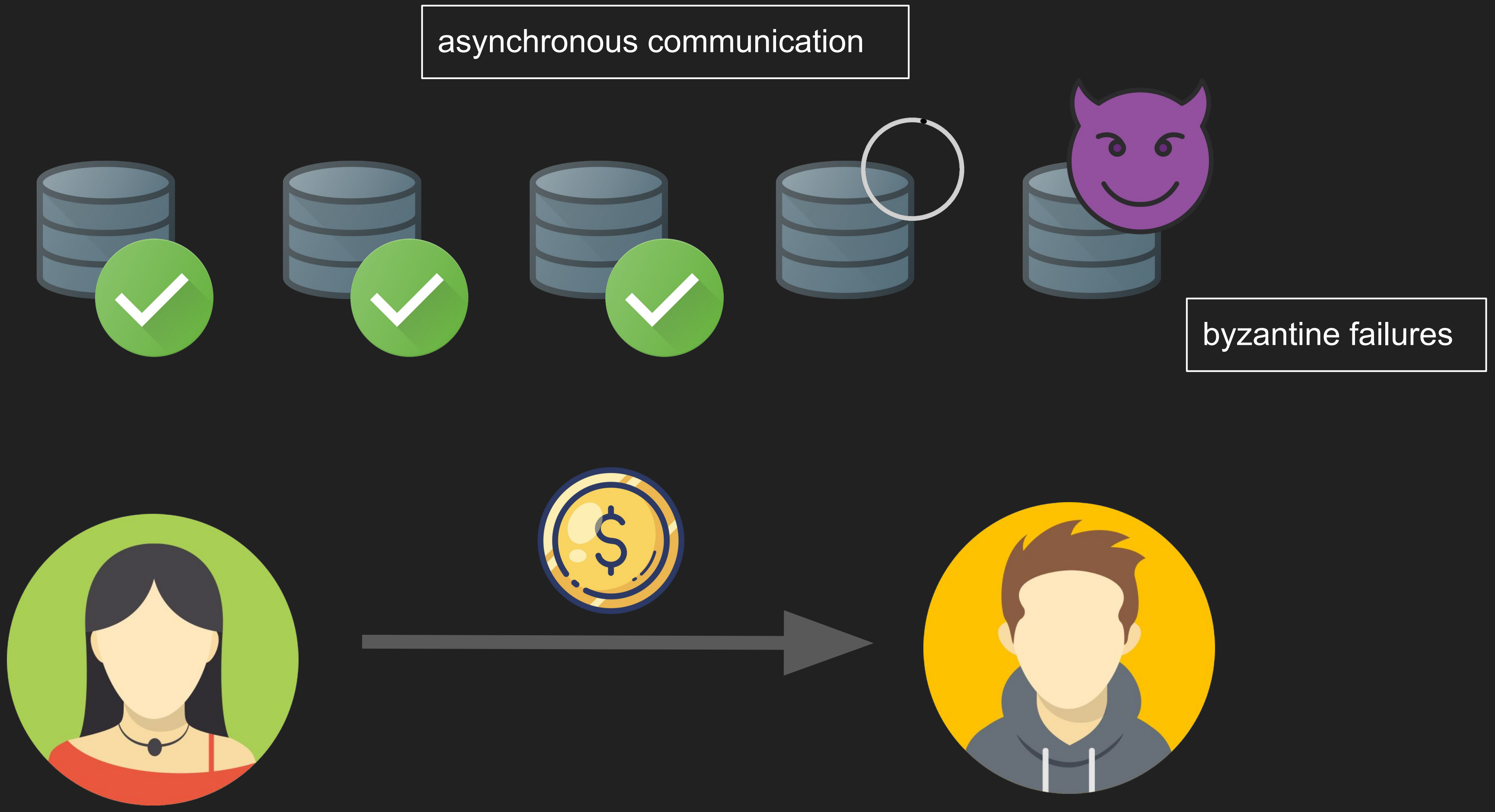
**Abstract.** There is a preconception that a blockchain needs consensus. But consensus is a powerful distributed property with a remarkably high price tag. So one may wonder whether consensus is at all needed.
We introduce a new blockchain architecture called ABC that functions despite not establishing consensus, and comes with an array of advantages: ABC is permissionless, deterministic, and resilient to complete asynchrony. ABC features finality and does not rely on costly proof-of-work.
Without establishing consensus, ABC cannot support certain applica-

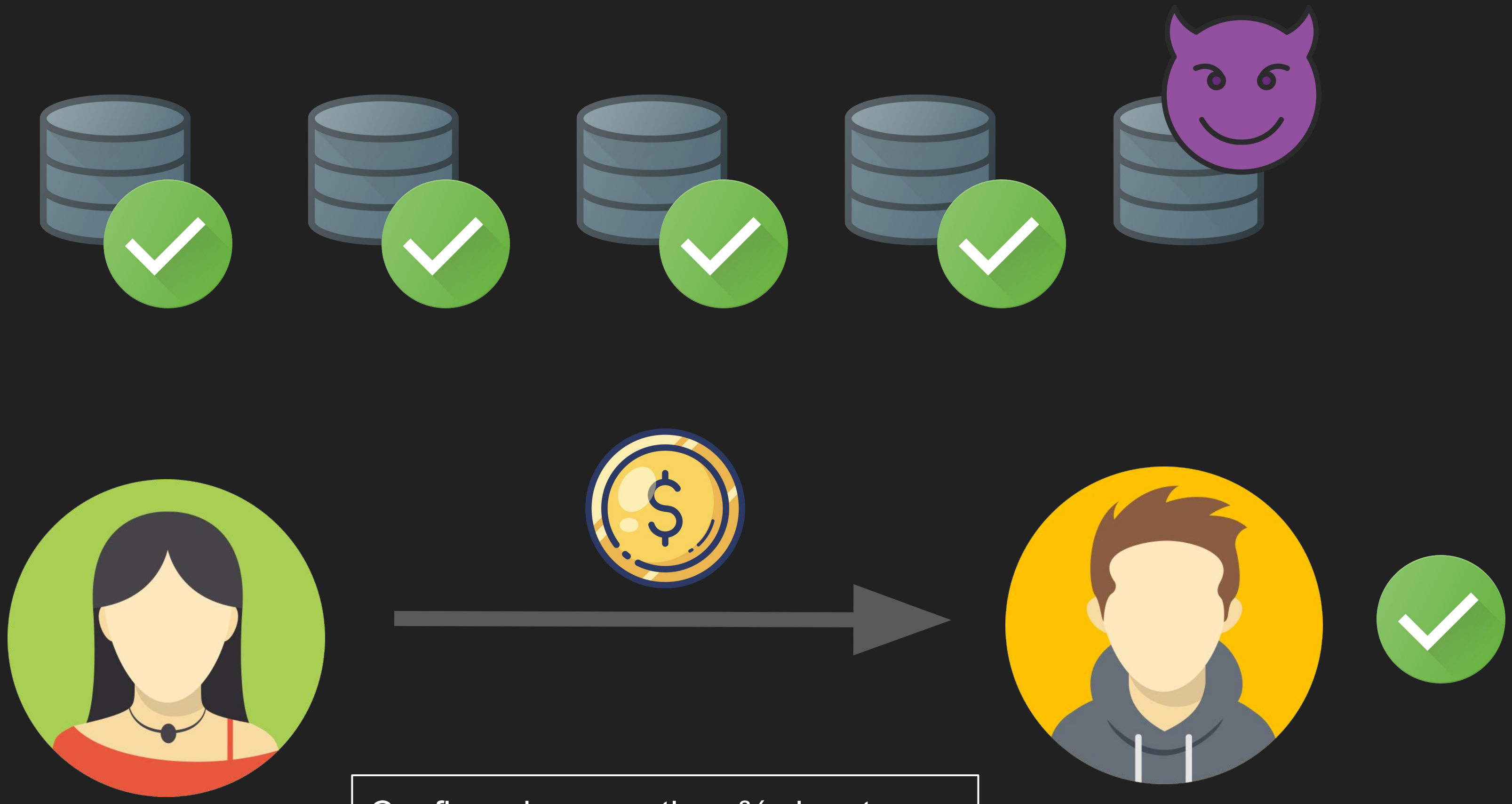




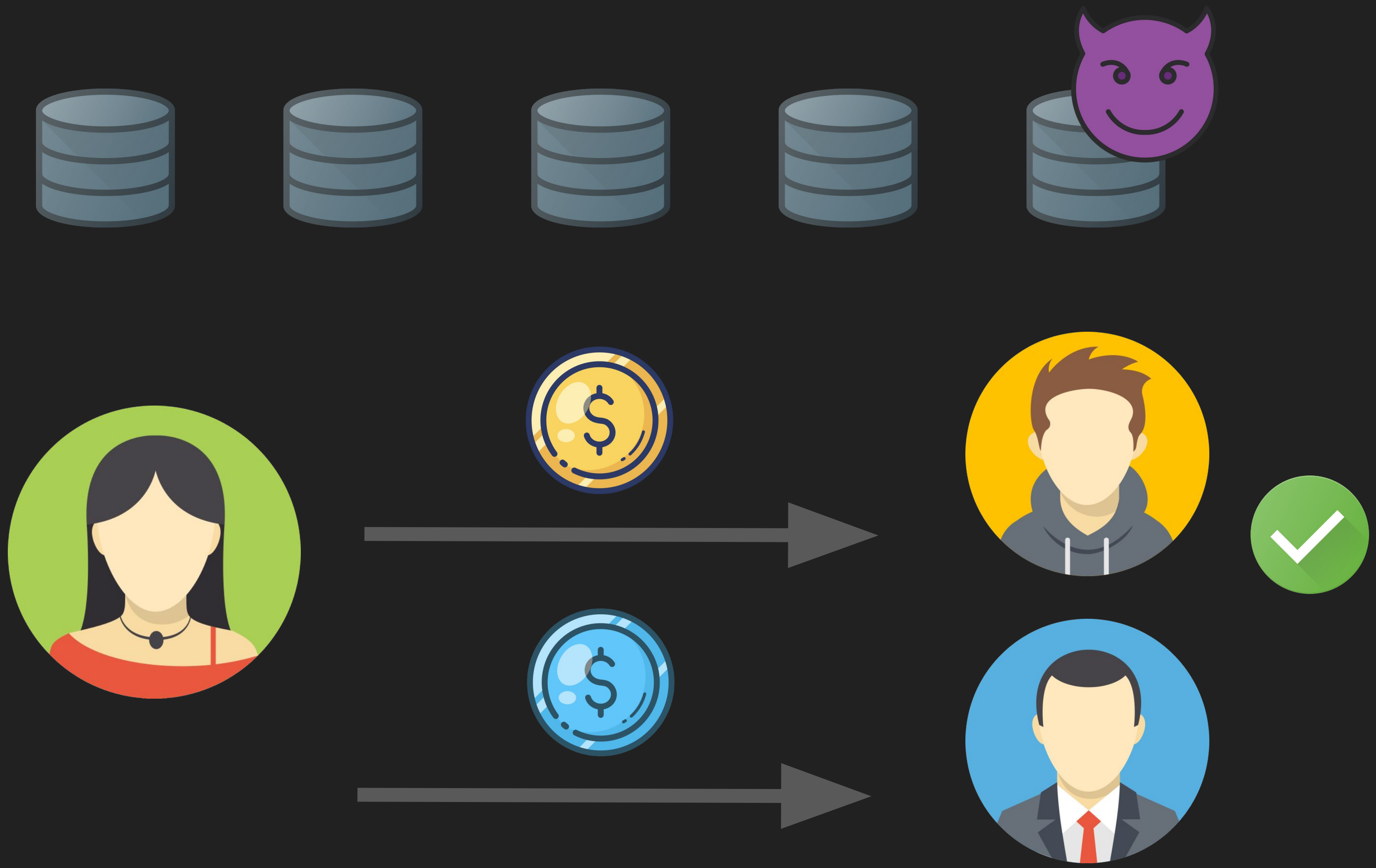


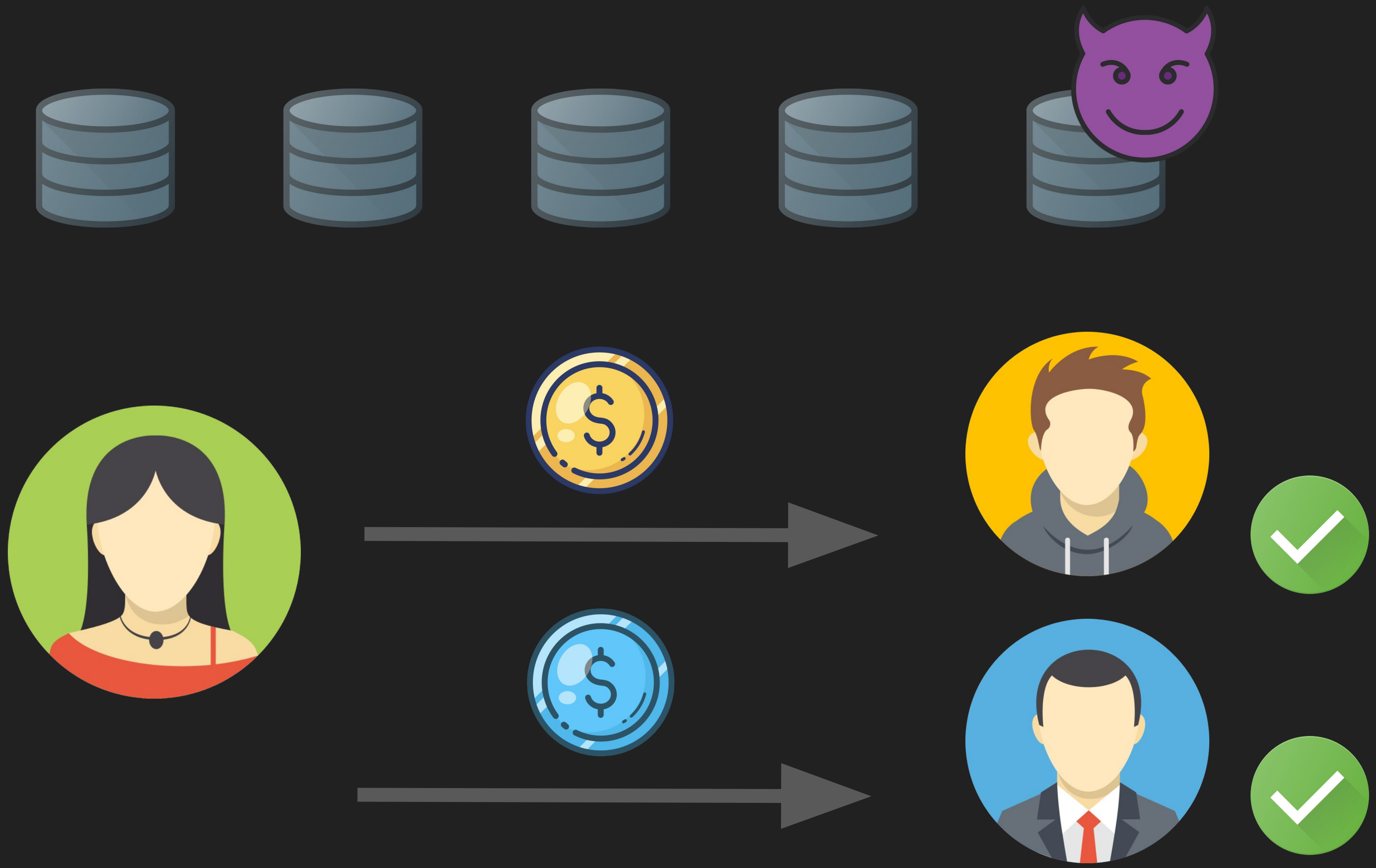



asynchronous communication
byzantine failures



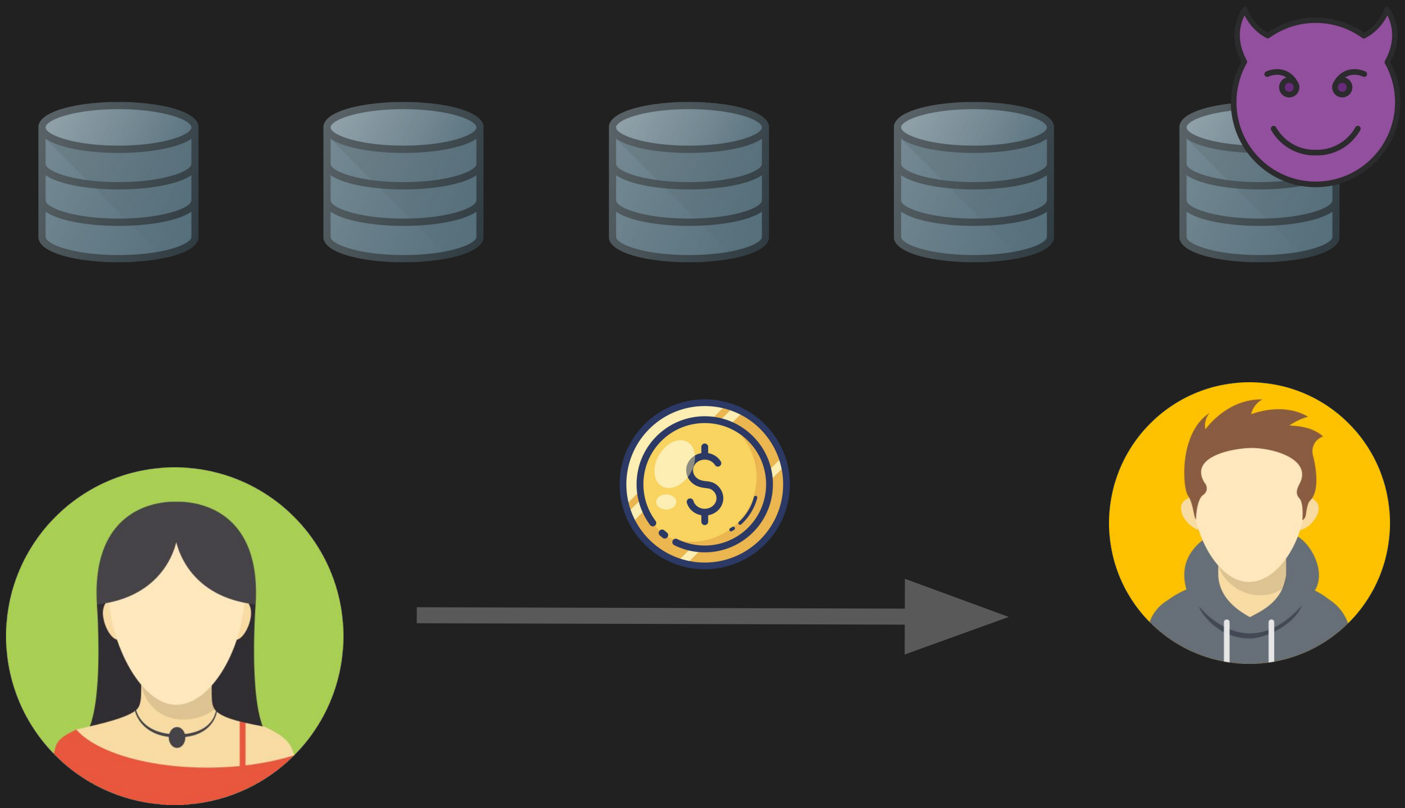
Confirmed = more than ⅔ signatures

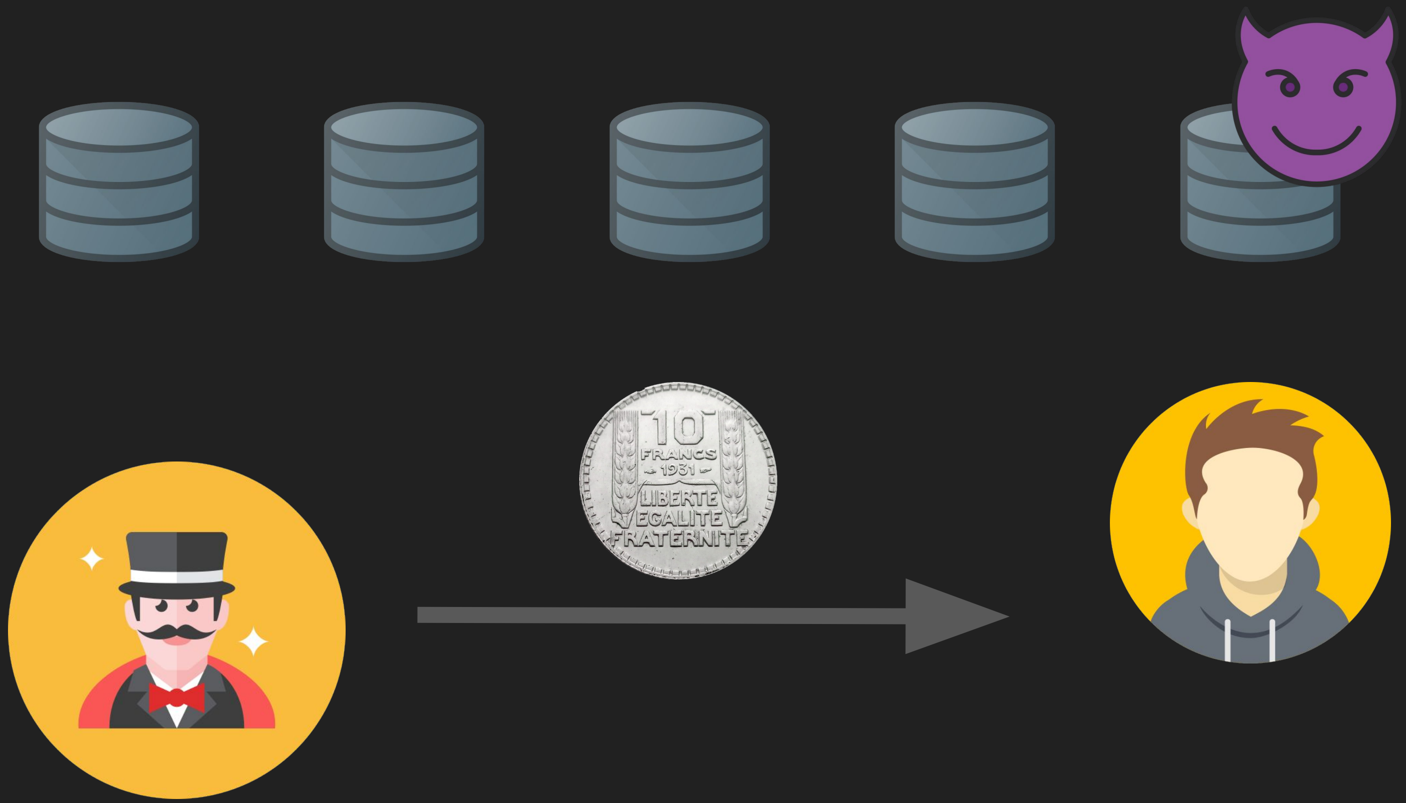


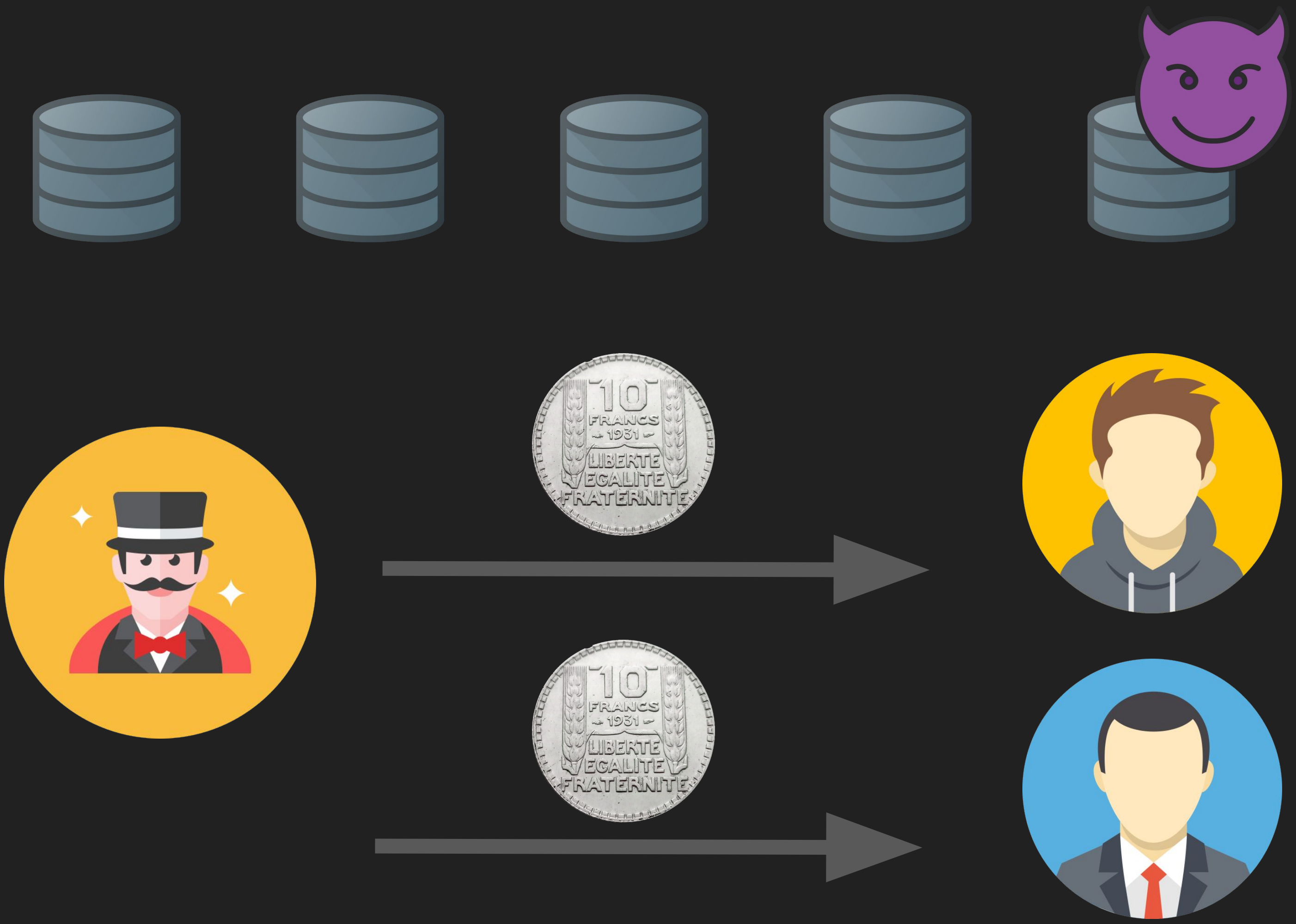


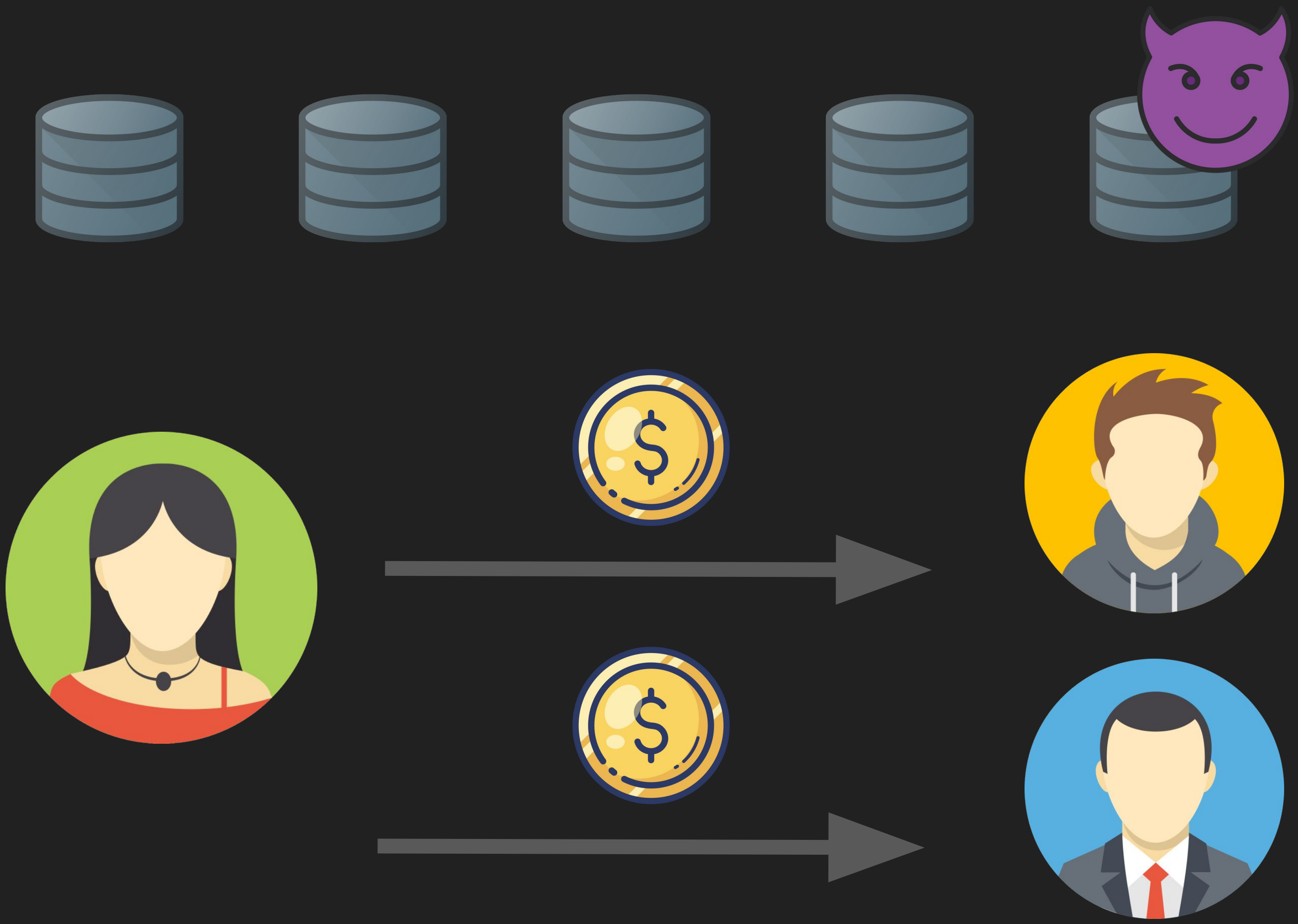
# What benefits do we have?

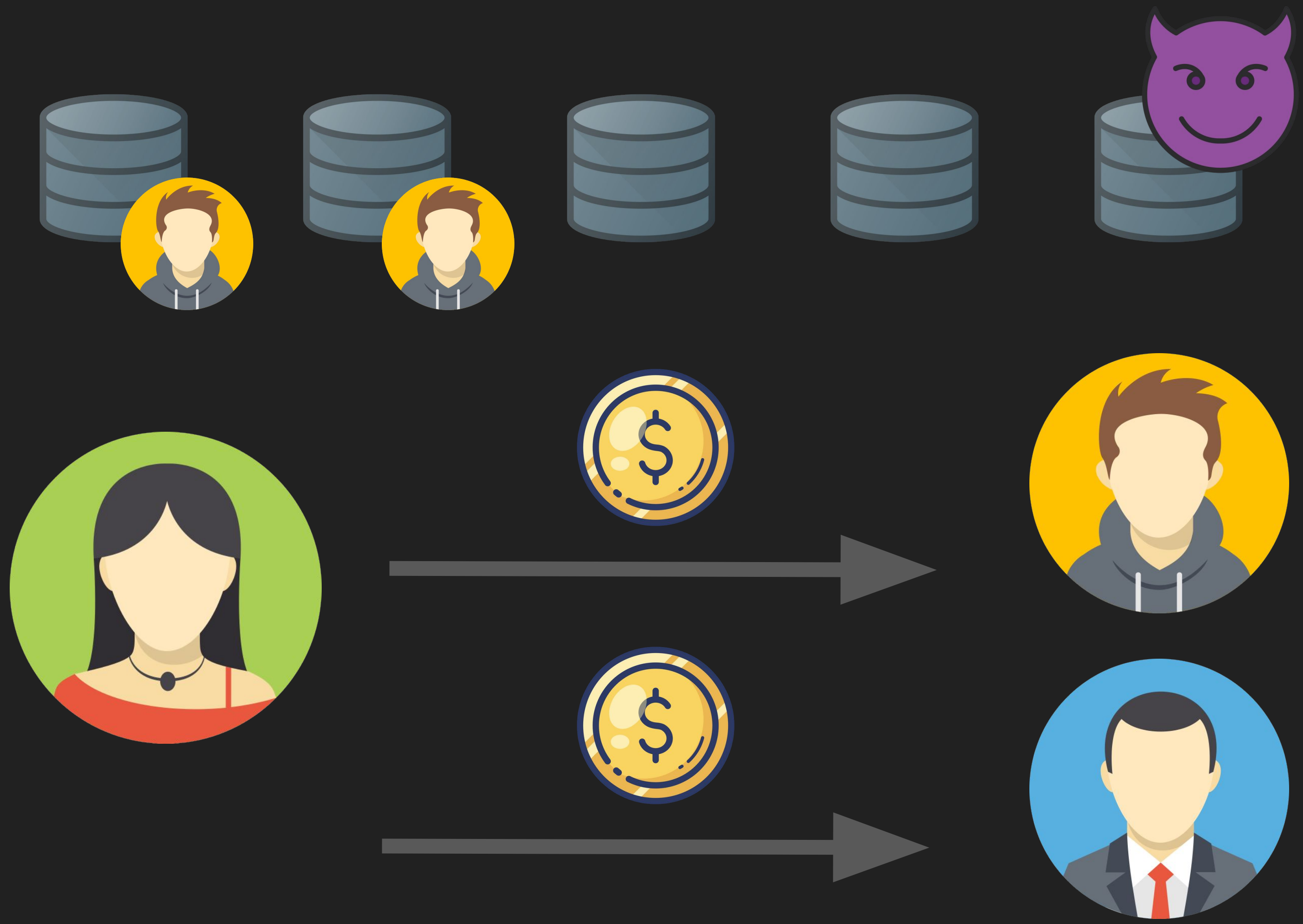
- Low latency
- Parallel execution →
  **Horizontal scaling**

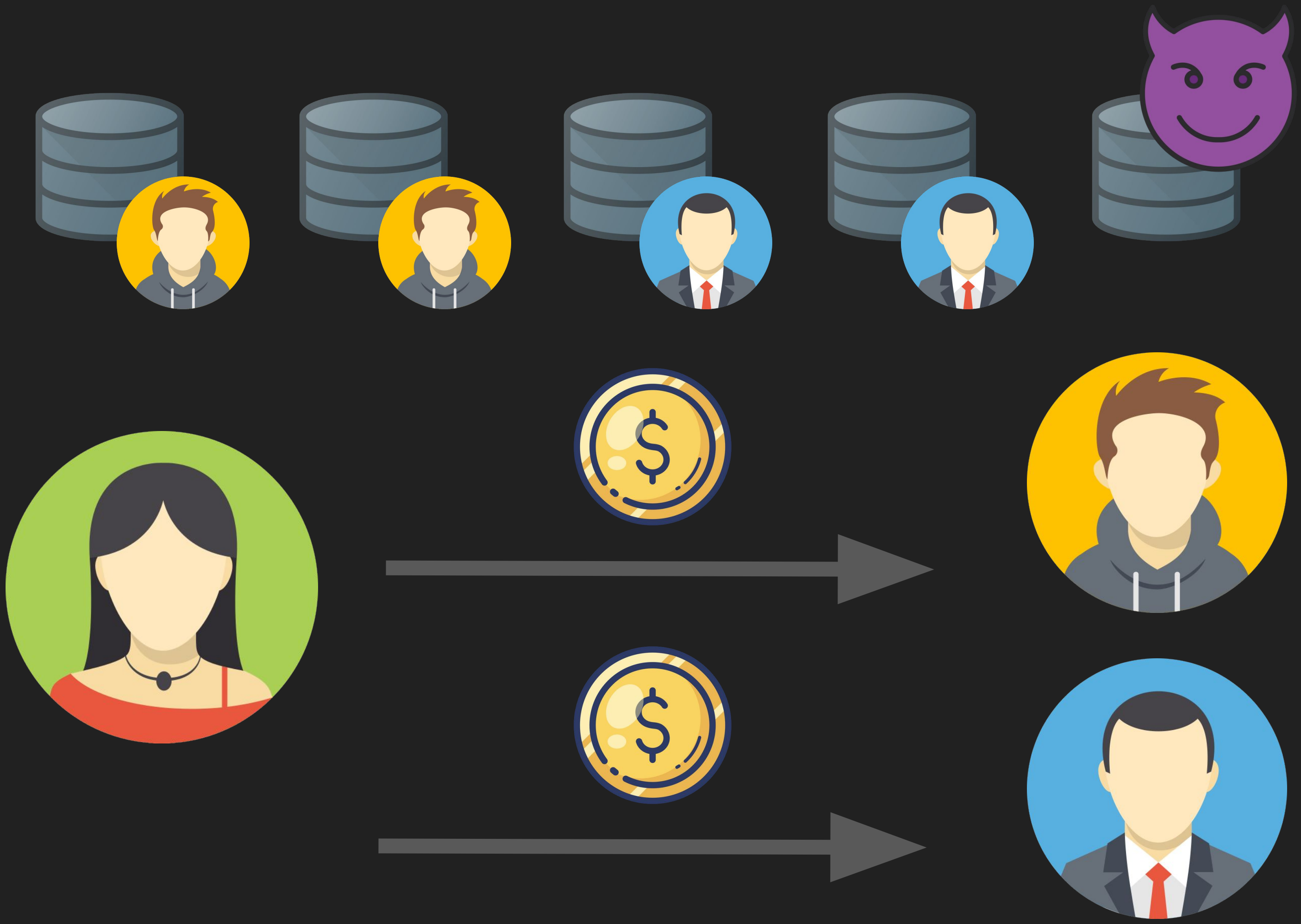


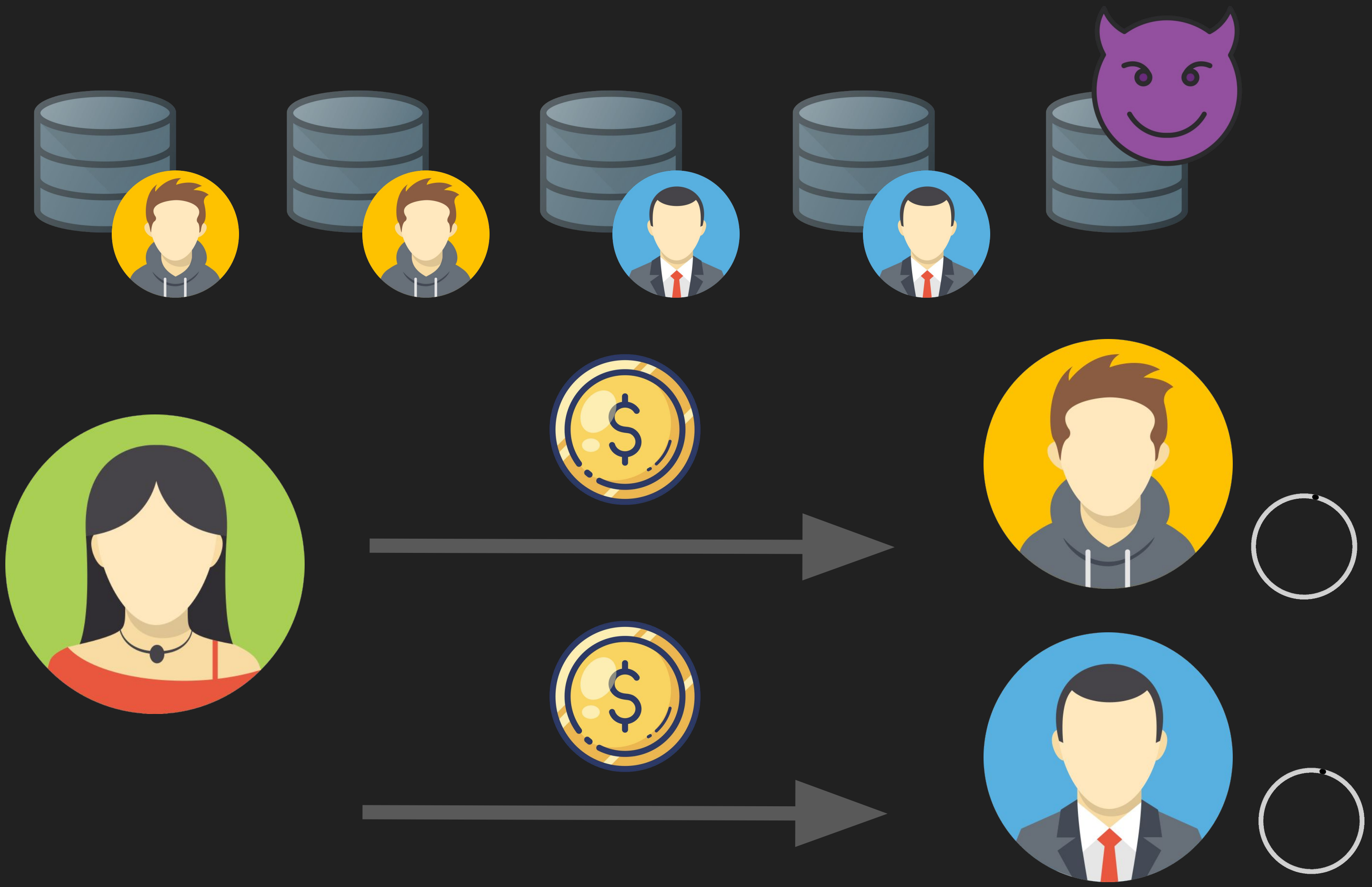

10
FRANCS
1931
LIBERTE
EGALITE
FRATERNITE


10
FRANCS
1931
LIBERTE
EGALITE
FRATERNITE
10
FRANCS
1931
LIBERTE
EGALITE
FRATERNITE







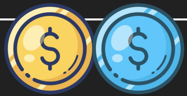



Transaction
A
Transaction
B
NO-CONSENSUS
JAKE-CLARK.TUMBLR
imgflip.com

What can we do with this system?

Verification

A specific type of online payments!

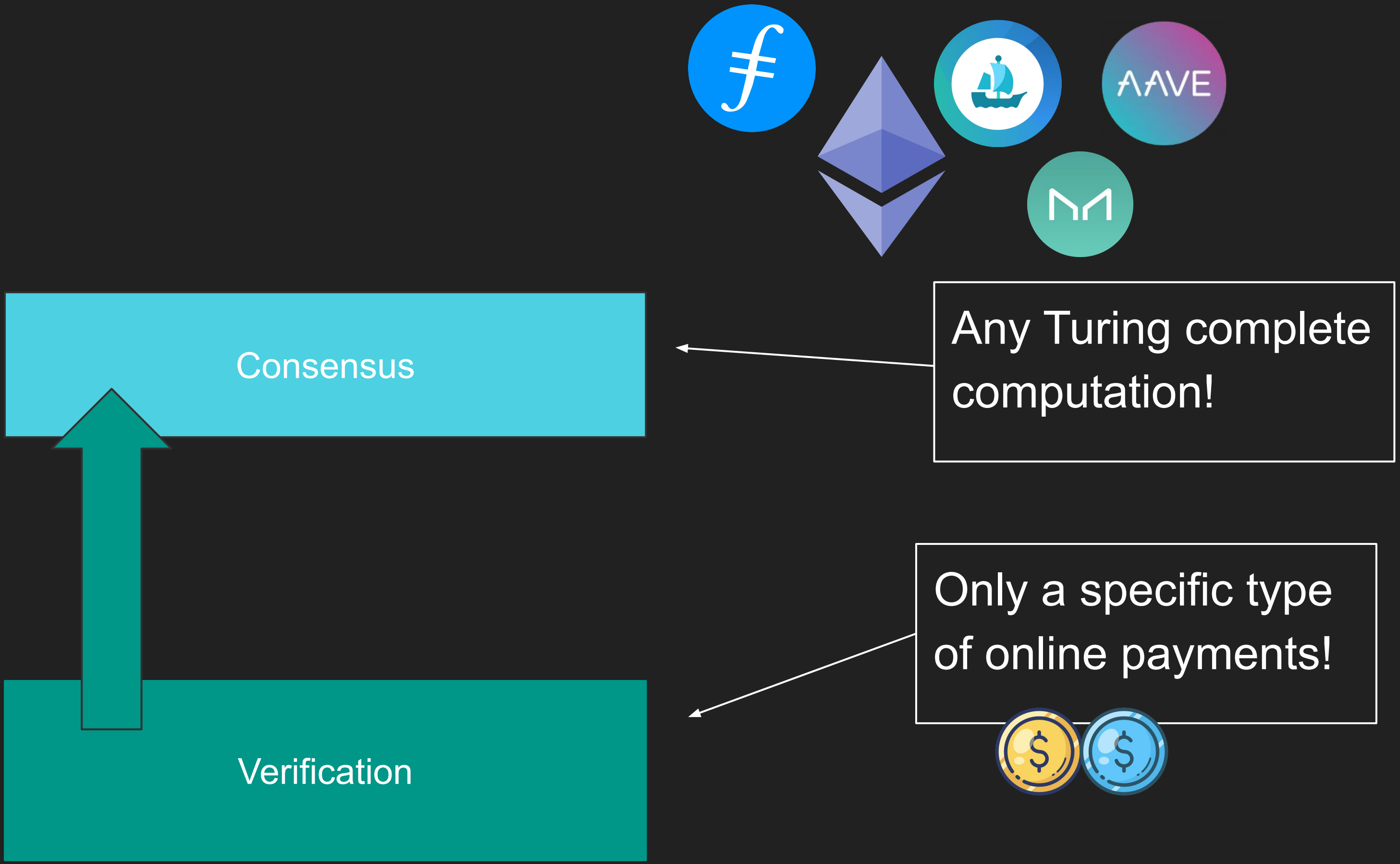


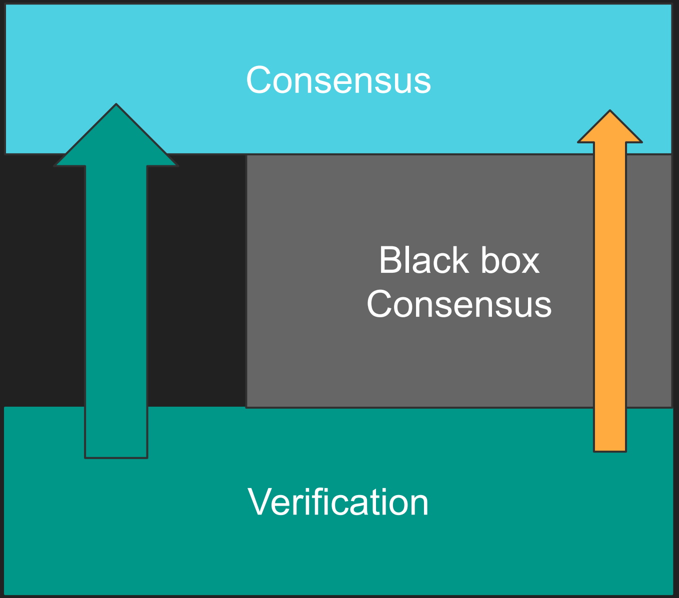
Consensus

Any Turing complete computation!

Verification

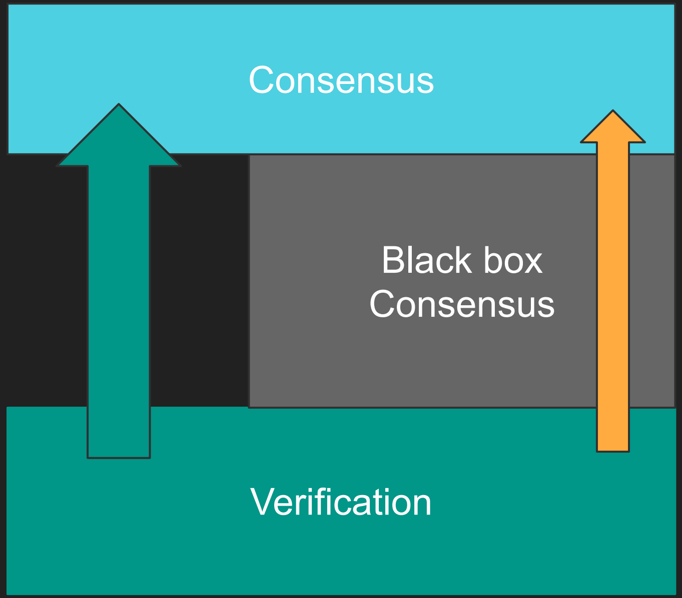
Only a specific type of online payments!


AAVE
Consensus
Verification
Any Turing complete computation!
Only a specific type of online payments!




Any Turing complete computation!

Only a specific type of online payments!




Only done when necessary!

# Consensus on Demand

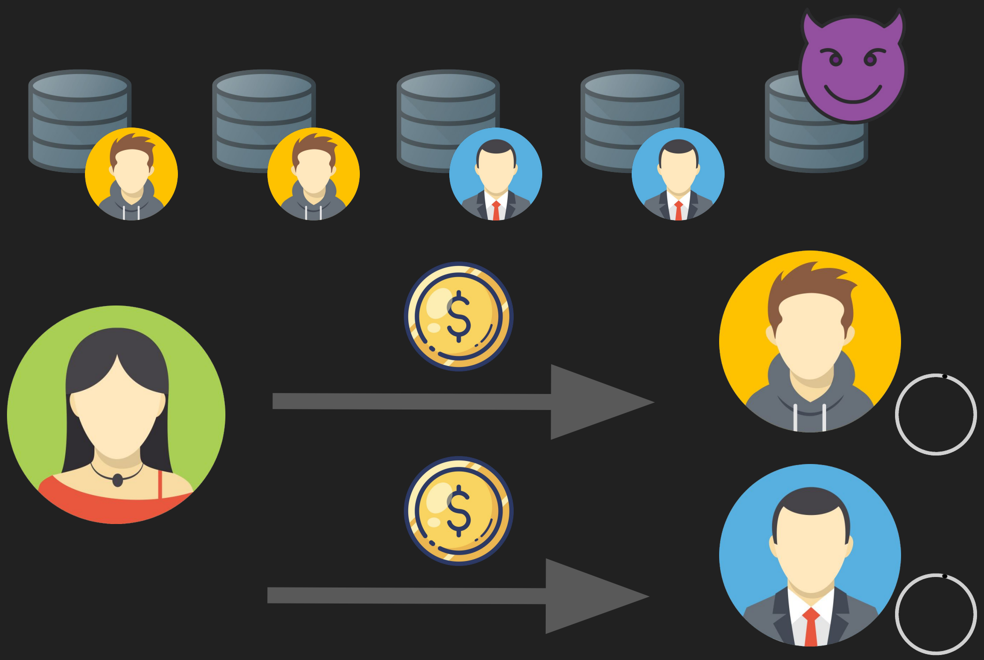
## Consensus on Demand

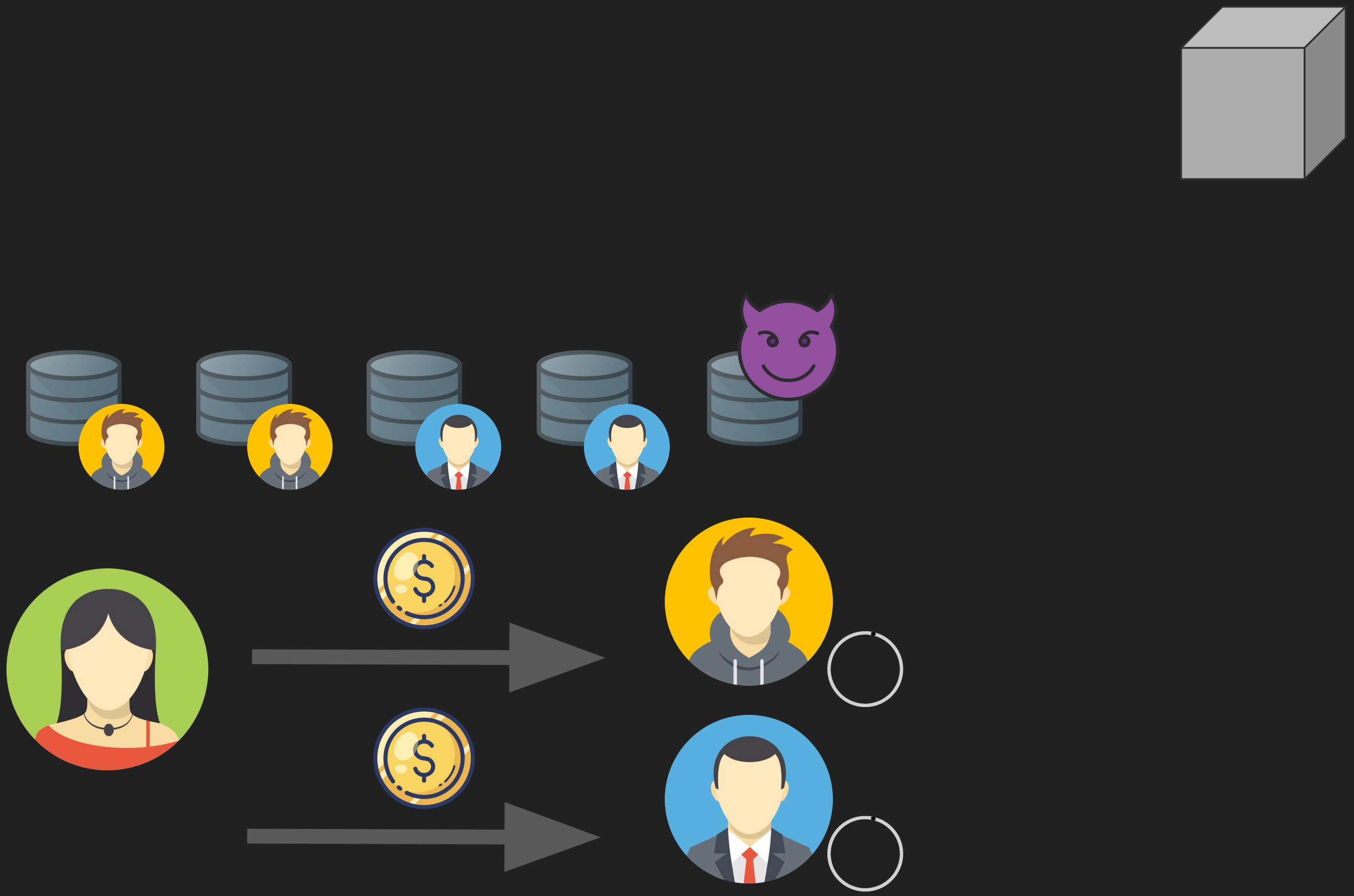
Jakub Sliwinski, Yann Vonlanthen, and Roger Wattenhofer

ETH Zurich, Switzerland

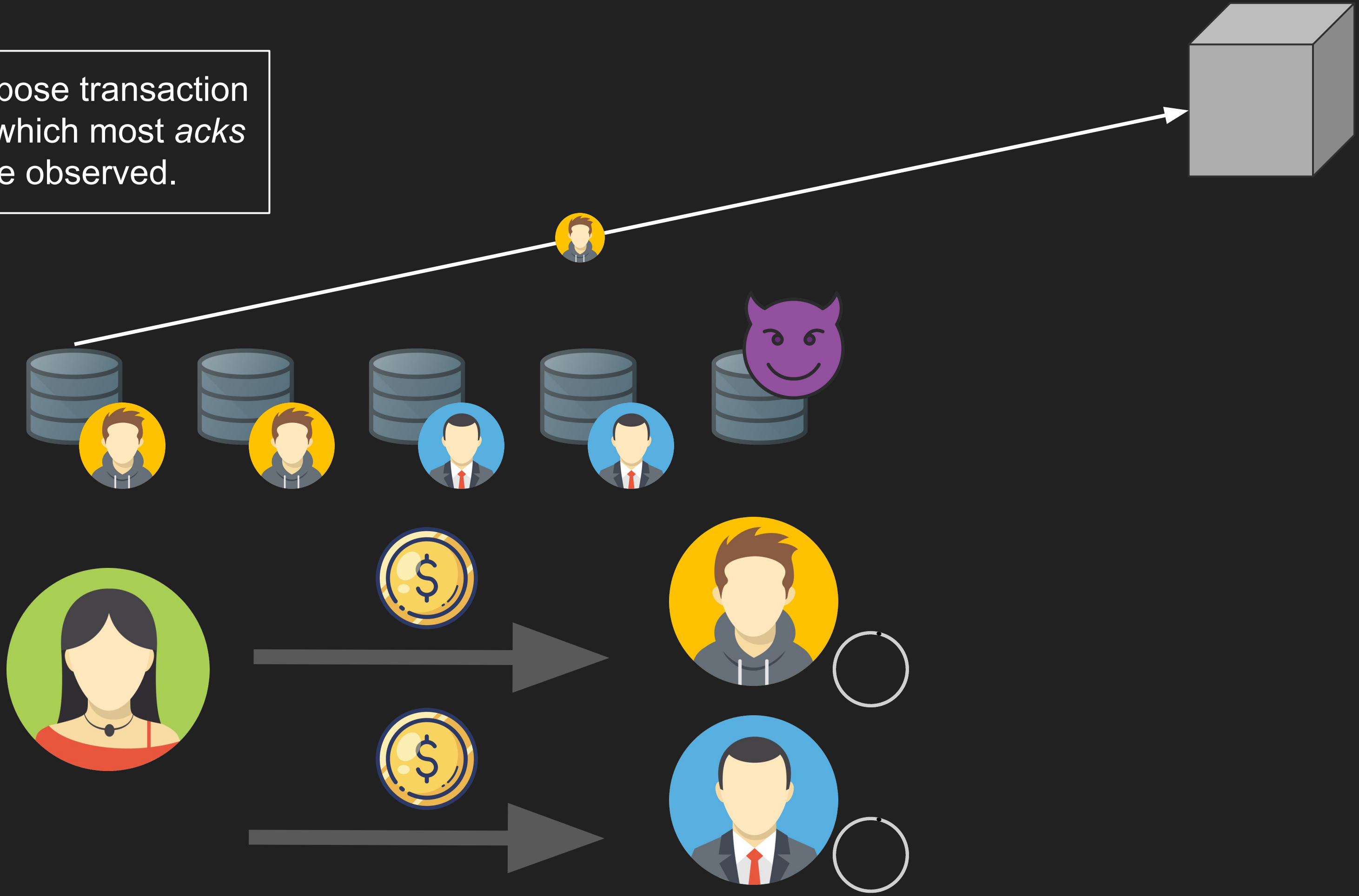
**Abstract.** Digital money can be implemented efficiently by avoiding consensus. However, no-consensus implementations have drawbacks, as they cannot support smart contracts, and (even more fundamentally) they cannot deal with conflicting transactions.
We present a novel protocol that combines the benefits of an asynchronous, broadcast-based digital currency, with the capacity to perform consensus. This is achieved by selectively performing consensus a posteriori, i.e., only when absolutely necessary. Our on-demand consensus comes at the price of restricting the Byzantine participants to be less than a one-fifth minority in the system, which is the optimal threshold. We formally prove the correctness of our system and present an open-source implementation, which inherits many features from the Ethereum ecosystem.

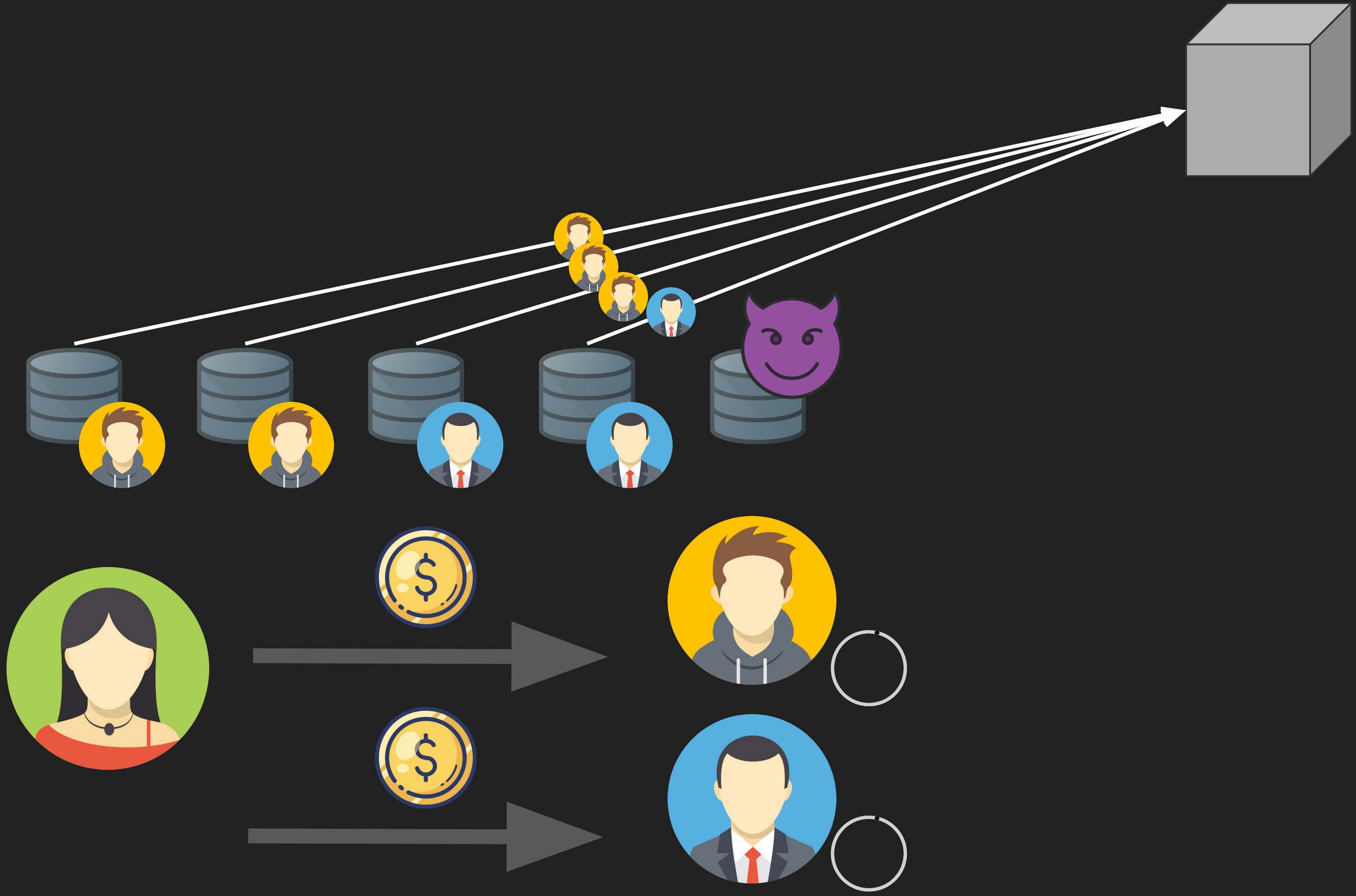
**Keywords:** Blockchain · Byzantine fault tolerance · Consensus · Cryptocurrencies · Reliable broadcast.

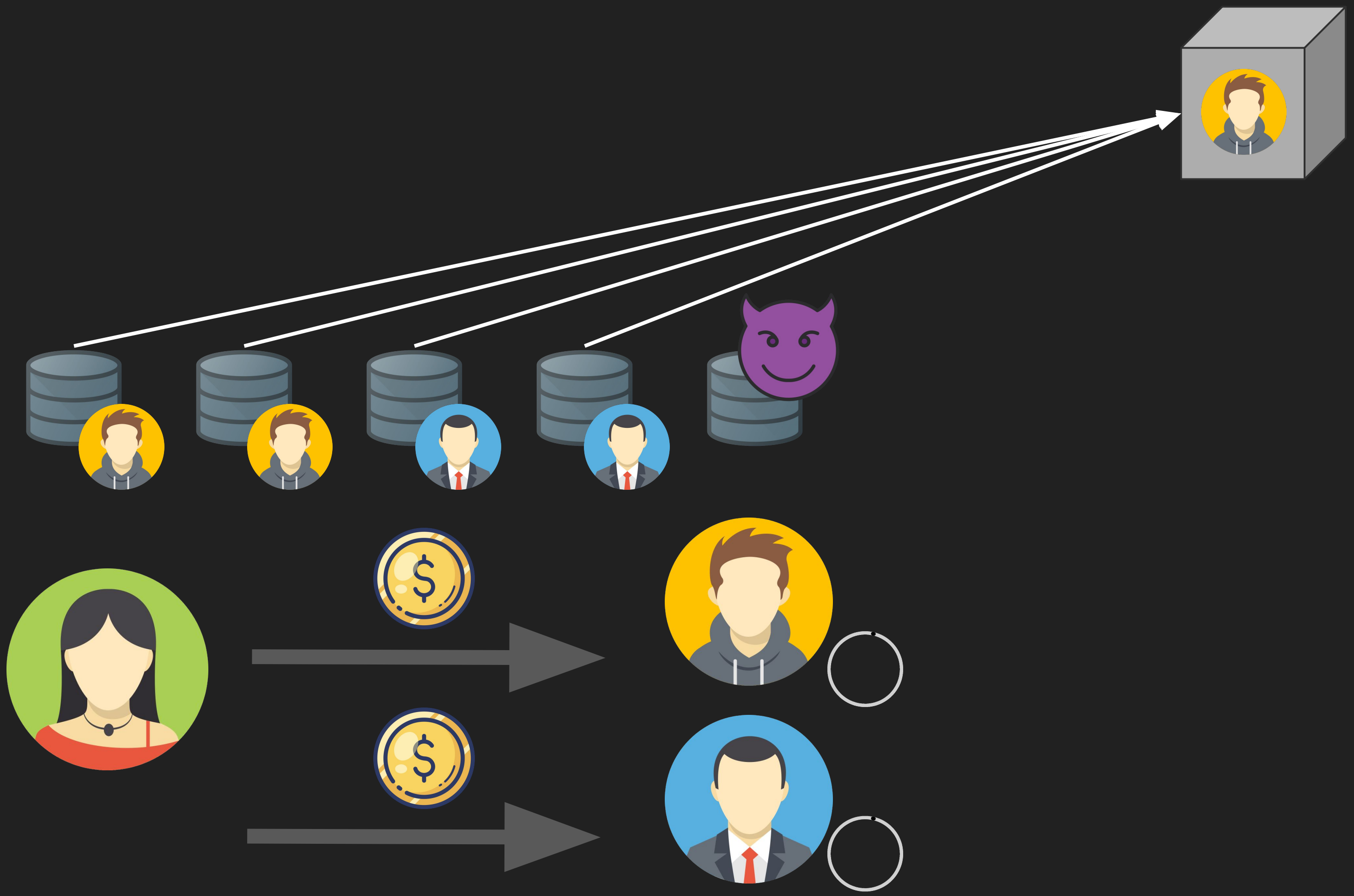


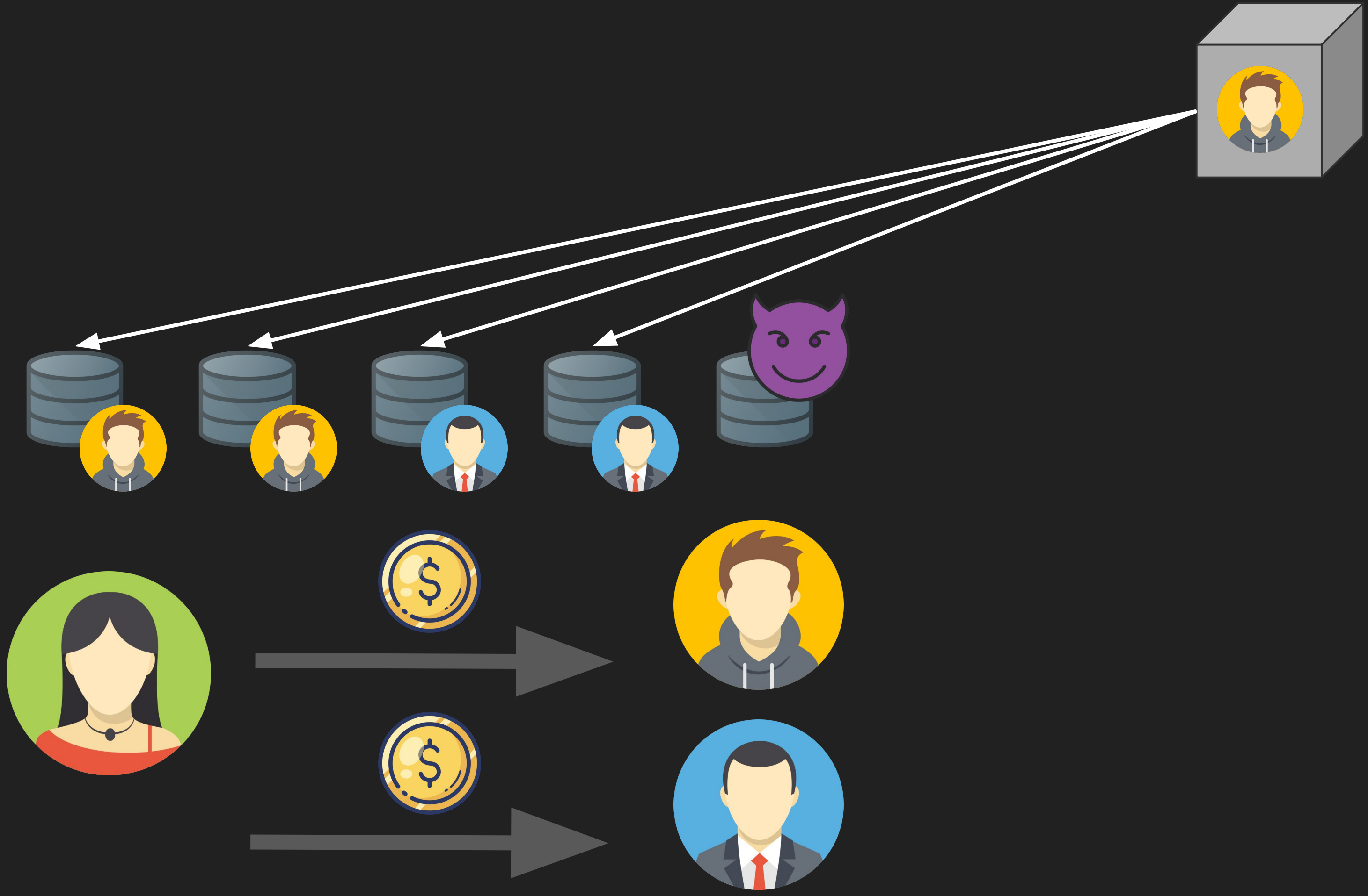


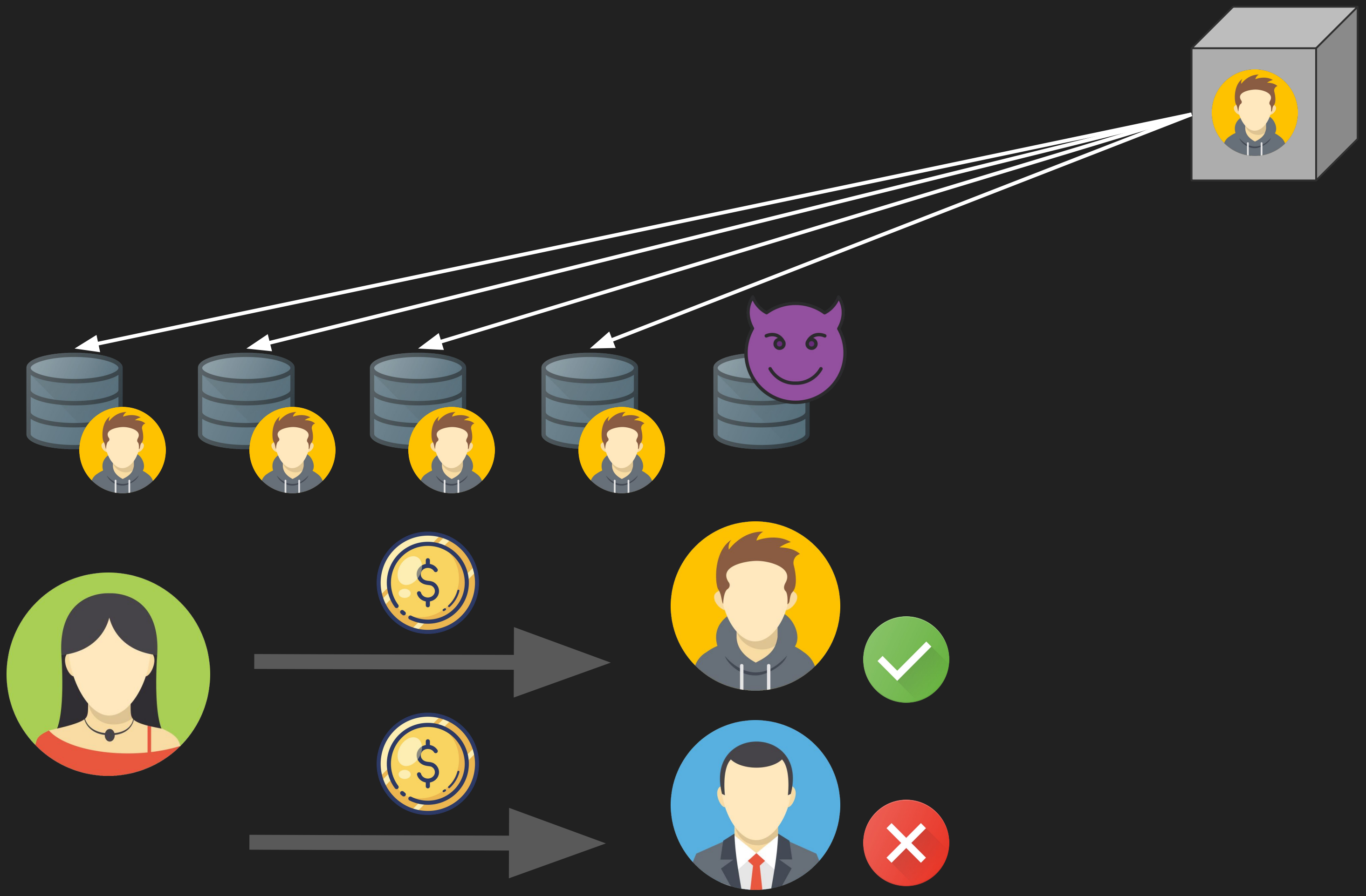
Propose transaction for which most *acks* were observed.

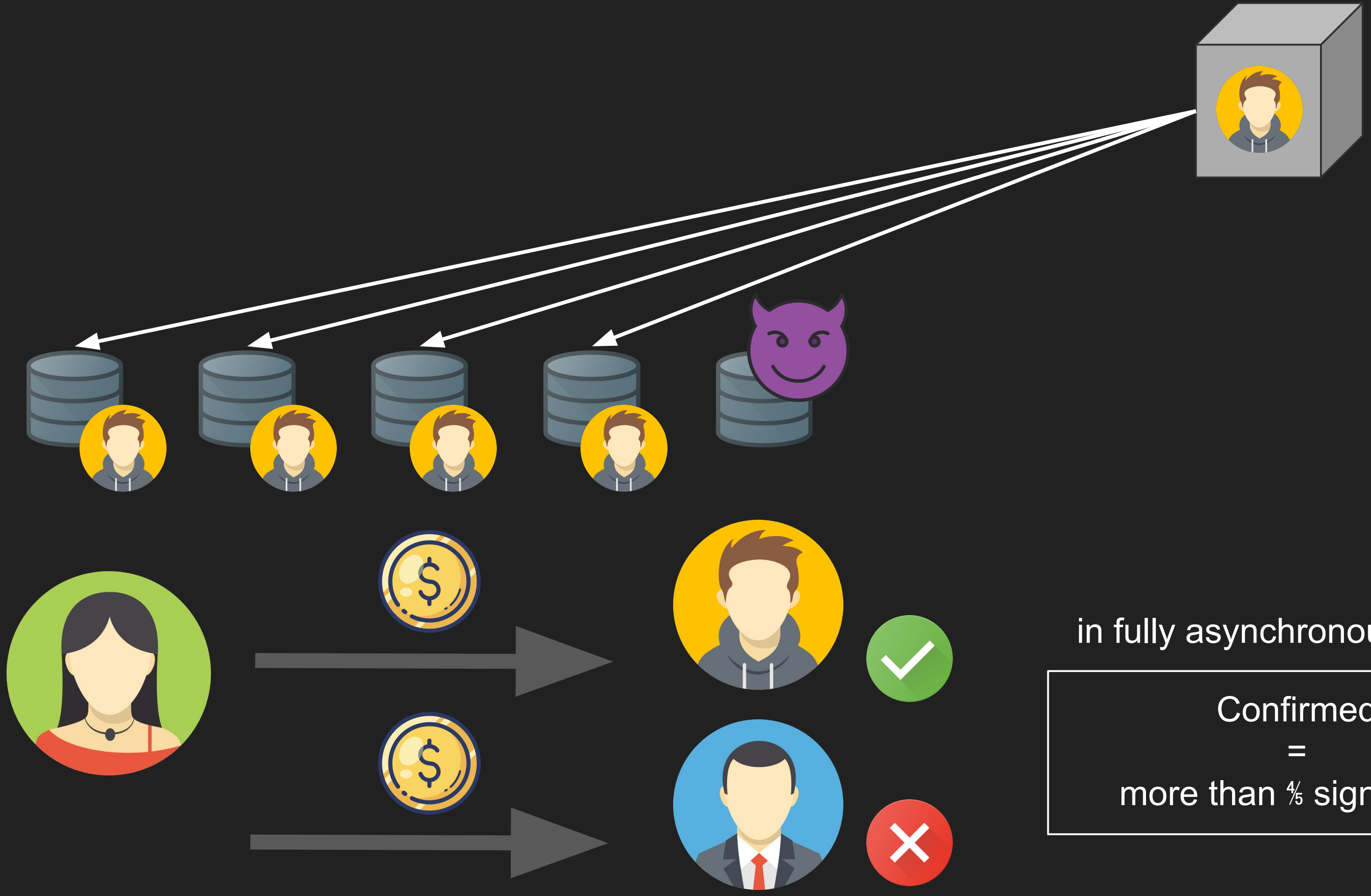








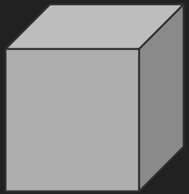




in fully asynchronous model:

Confirmed
=
more than ⅘ signatures

# The Best of Both Worlds



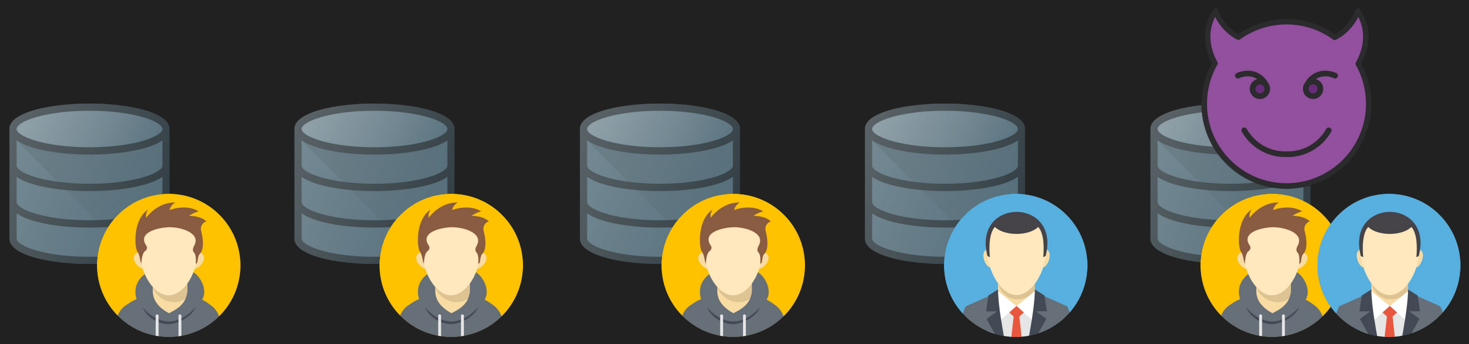
## Fast Path

- Low latency
- Parallelization
- + Lower message complexity
  - No need for BRB!

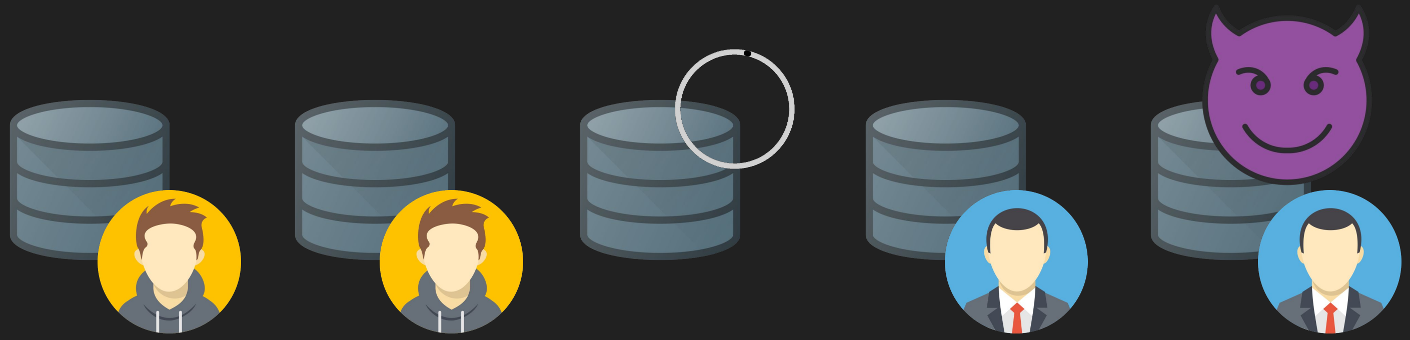
## Consensus

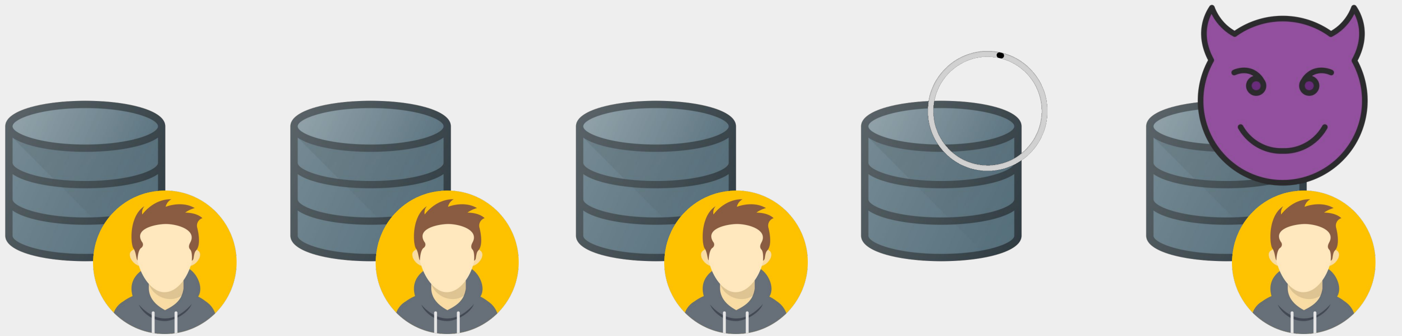
- Arbitrary Turing complete computation
  - Enables smart contracts

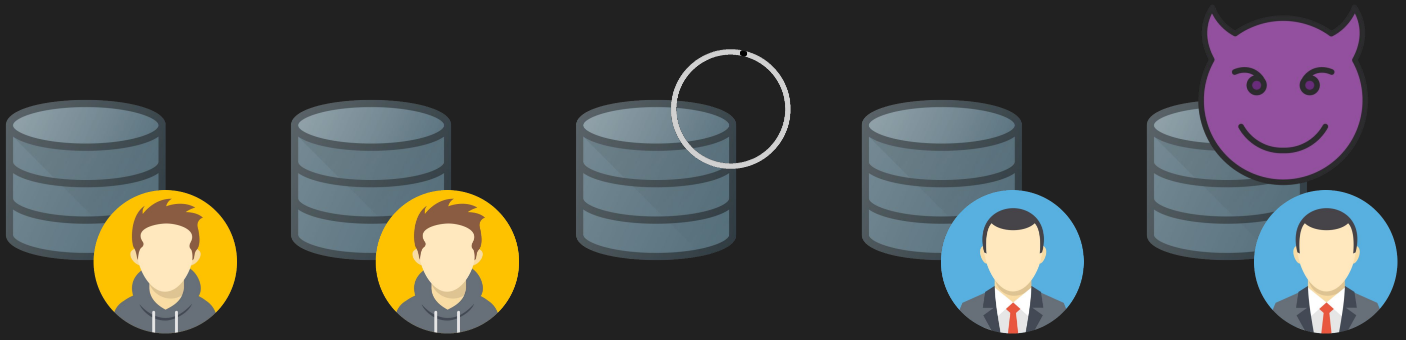
Why do we need **n ≥ 5f + 1**?



Let’s assume n = 5f

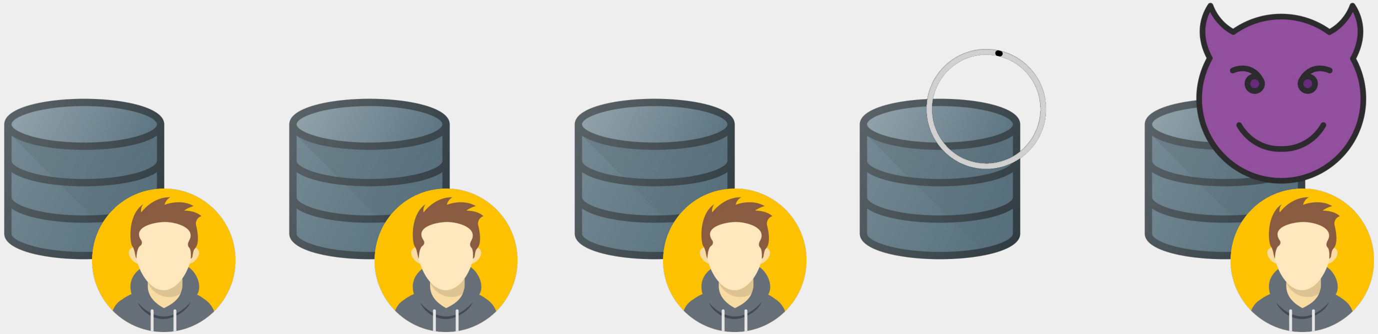




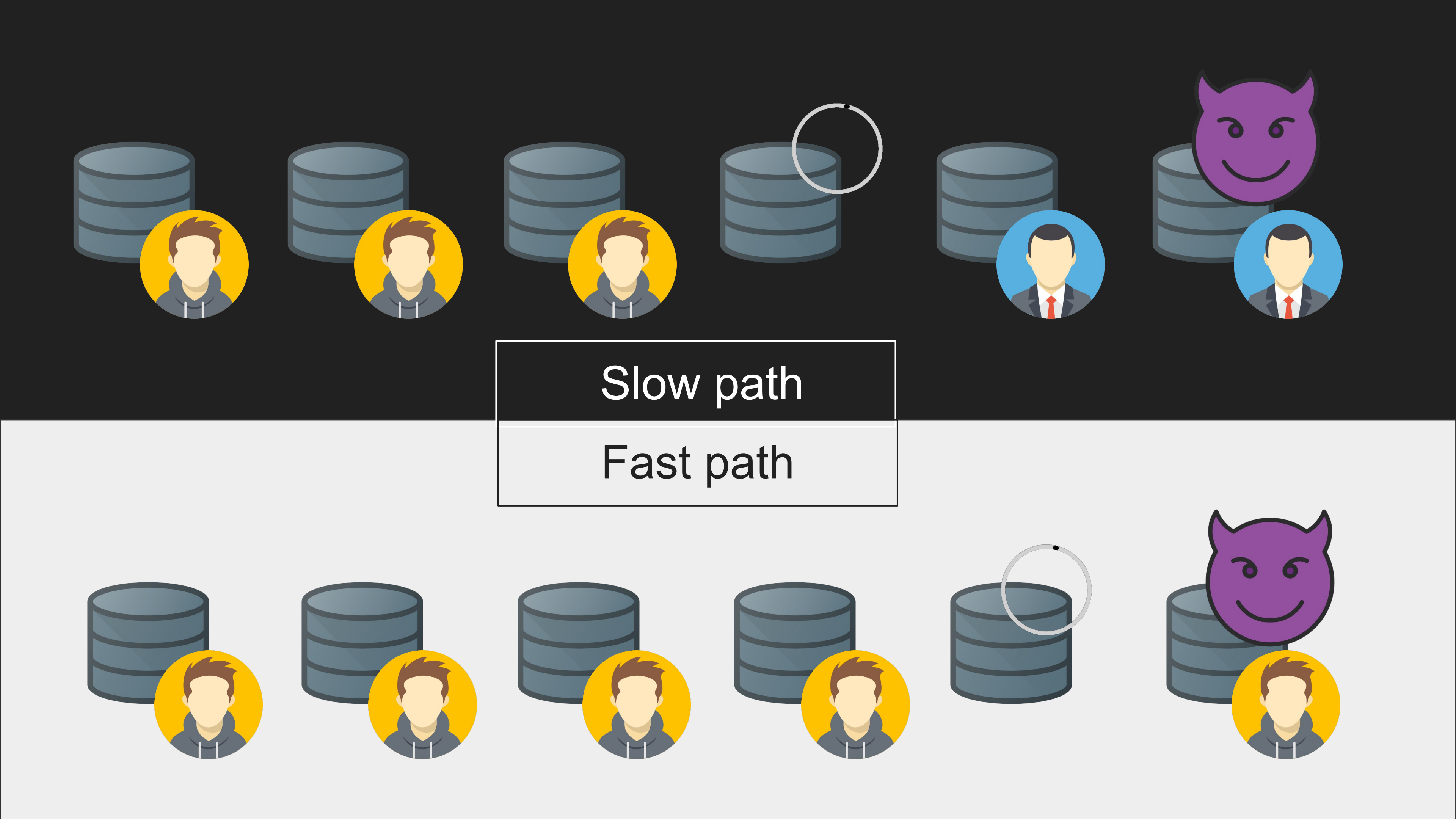


Slow path must agree

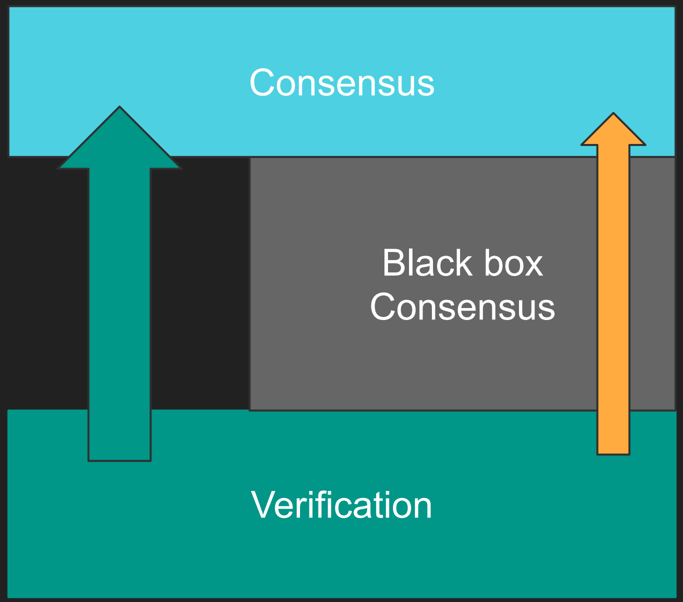
with fast path!



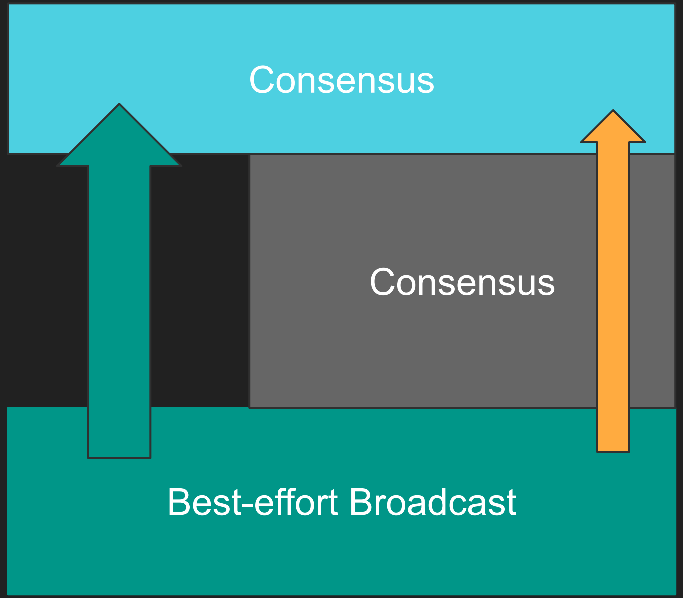
# How can we guarantee agreement?


Slow path
Fast path

# A modular approach, different alternatives exist!

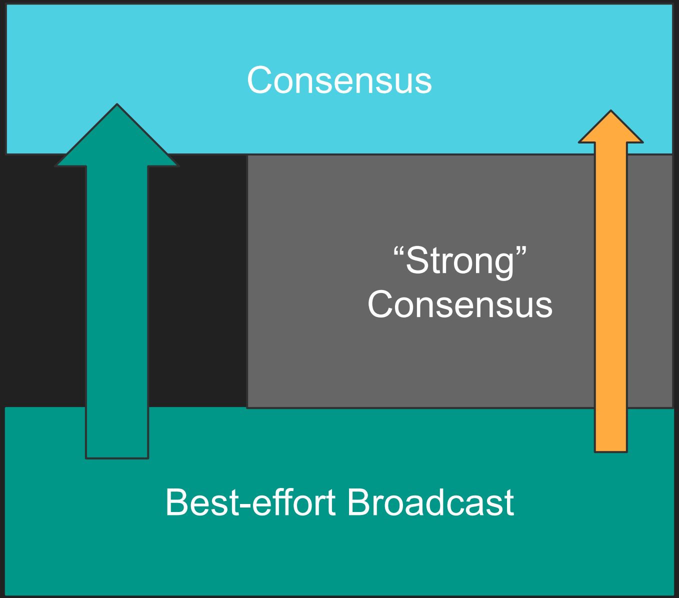


# A modular approach, different alternatives exist!



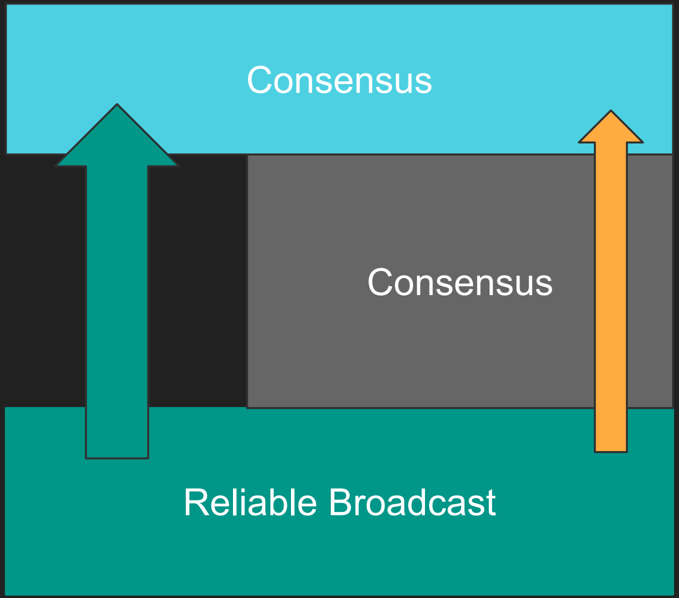

In case of any conflict, propose most observed *transaction* after (n+3f)/2 observed *acks.*

# A modular approach, different alternatives exist!



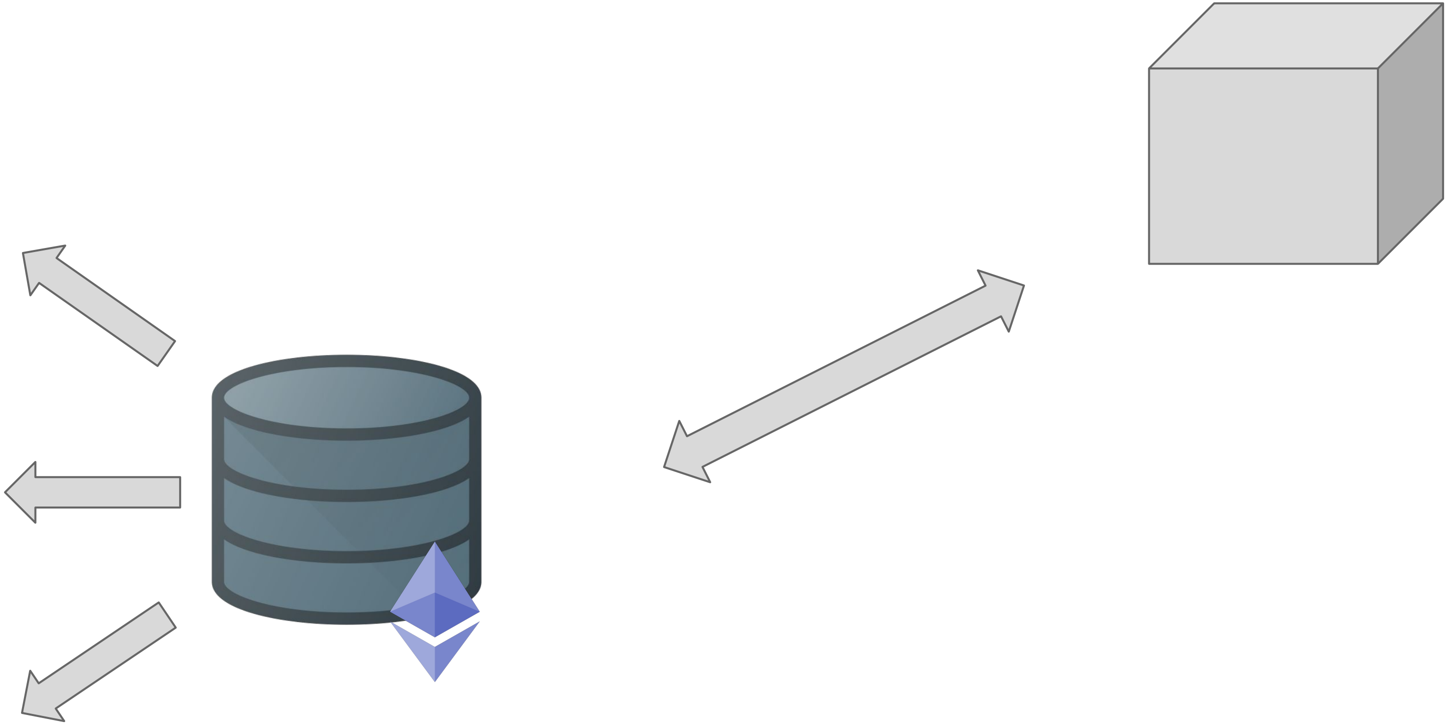

Propose own *ack* after observing a *conflict.*

# A modular approach, different alternatives exist!

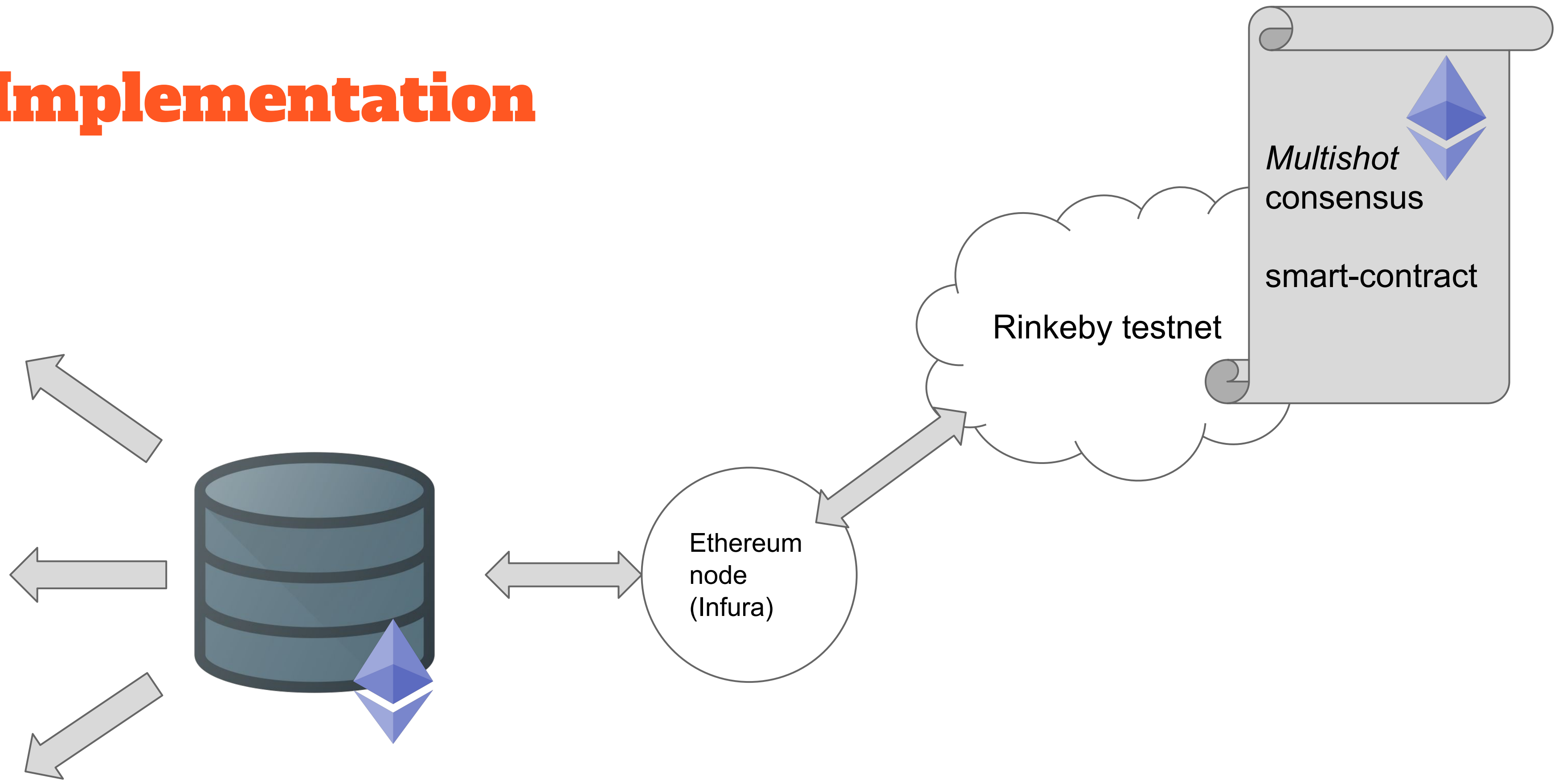


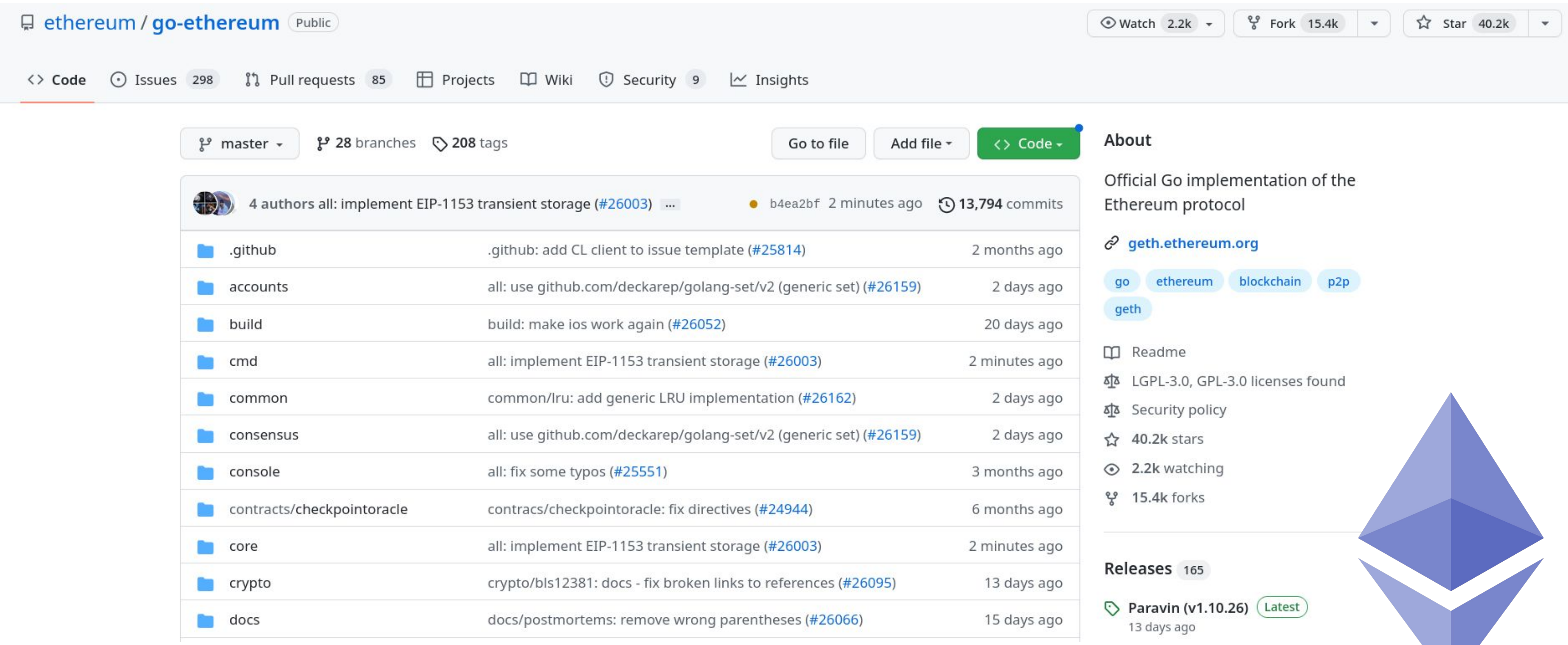

No need to participate in Consensus if fast path succeeds.

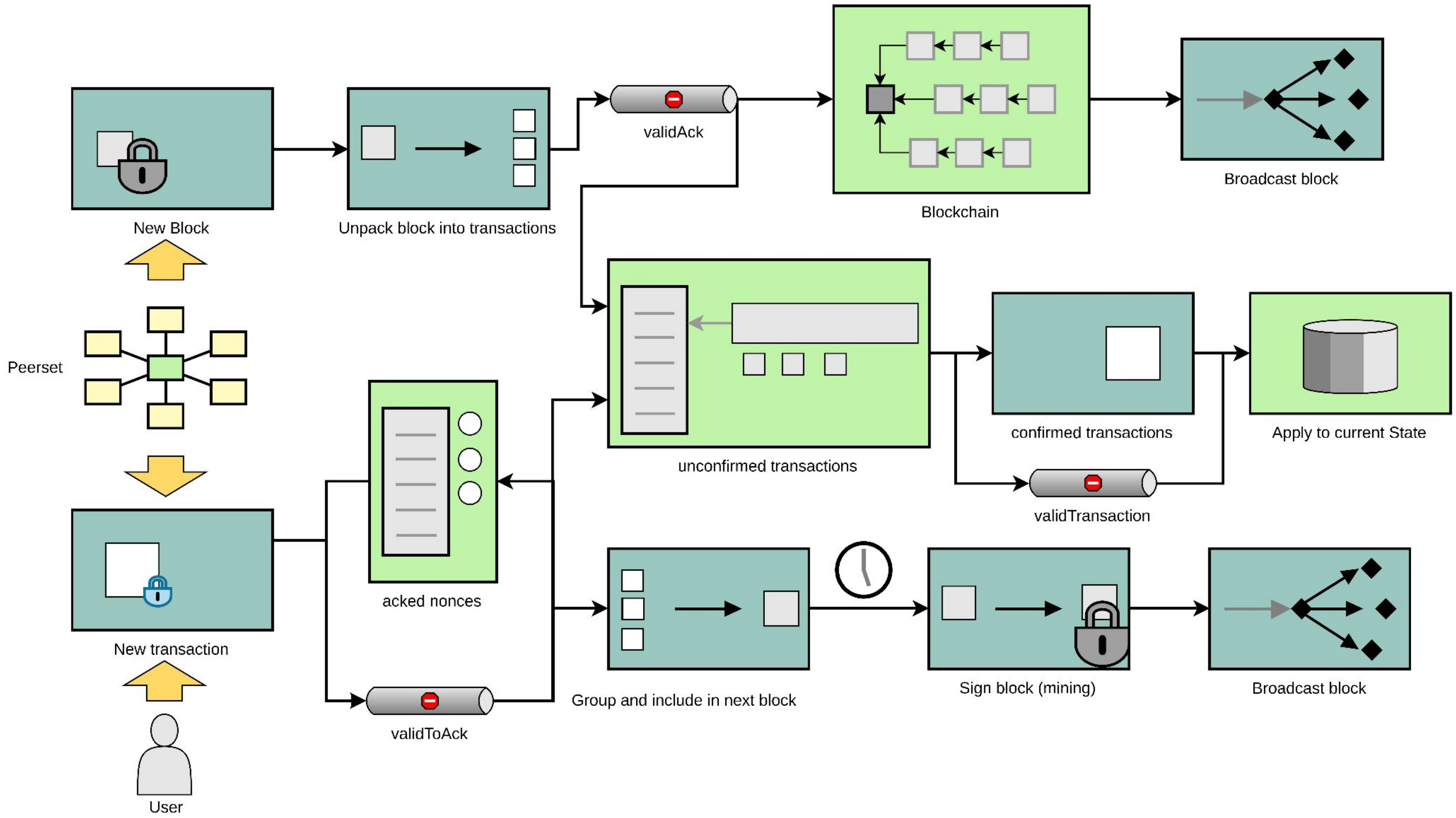
# Implementation



# Implementation




ethereum / go-ethereum Public
Watch 2.2k
Fork 15.4k
Star 40.2k
Code
Issues 298
Pull requests 85
Projects
Wiki
Security 9
Insights
master
28 branches
208 tags
Go to file
Add file
Code
4 authors all: implement EIP-1153 transient storage (#26003) ...
b4ea2bf 2 minutes ago
13,794 commits
.github
.github: add CL client to issue template (#25814)
2 months ago
accounts
all: use github.com/deckarep/golang-set/v2 (generic set) (#26159)
2 days ago
build
build: make ios work again (#26052)
20 days ago
cmd
all: implement EIP-1153 transient storage (#26003)
2 minutes ago
common
common/lru: add generic LRU implementation (#26162)
2 days ago
consensus
all: use github.com/deckarep/golang-set/v2 (generic set) (#26159)
2 days ago
console
all: fix some typos (#25551)
3 months ago
contracts/checkpointoracle
contracs/checkpointoracle: fix directives (#24944)
6 months ago
core
all: implement EIP-1153 transient storage (#26003)
2 minutes ago
crypto
crypto/bls12381: docs - fix broken links to references (#26095)
13 days ago
docs
docs/postmortems: remove wrong parentheses (#26066)
15 days ago
About
Official Go implementation of the Ethereum protocol
geth.ethereum.org
go
ethereum
blockchain
p2p
geth
Readme
LGPL-3.0, GPL-3.0 licenses found
Security policy
40.2k stars
2.2k watching
15.4k forks
Releases 165
Paravin (v1.10.26) Latest
13 days ago


New Block
Unpack block into transactions
validAck
Blockchain
Broadcast block
Peerset
unconfirmed transactions
confirmed transactions
Apply to current State
validTransaction
acked nonces
New transaction
User
validToAck
Group and include in next block
Sign block (mining)
Broadcast block

| | Bitcoin and Ethereum [32] | Ouroboros [24] | Algorand [19] | PBFT [12] | Red Belly [15] | BEAT [17] | Broadcast-based [14] | CoD + PBFT | CoD + BEAT |
|---|---|---|---|---|---|---|---|---|---|
| Energy-efficient | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Deterministic finality | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Permissionless | ✓ | ✓ | ✓ | | | | | | |
| Leaderless | | | | | ✓ | ✓ | ✓ | | ✓ |
| Asynchronous | | | | | | ✓ | ✓ | | ✓ |
| Parallelizable | | | | | | | ✓ | ✓ | ✓ |
| Consensus | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ |

We provide a wrapper that minimizes the accesses to Consensus!

# Thank you! Merci!

yvonlanthen@ethz.ch