Chainspace:
A Sharded Smart Contract Platform

Authors
Mustafa Al-Bassam[*]
Alberto Sonnino[*]
Shehar Bano[*]
Dave Hrycyszyn[†]
George Danezis[*]

CHAINSPACE

Published in NDSS ‘18

[*] University College London
[†] constructiveproof.com

Binary District Meetup, London

Motivation

- Blockchains are cool — but scale badly

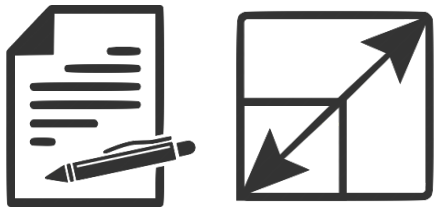
- Hard to operate on secret inputs

Introduction

- **What is chainspace?**

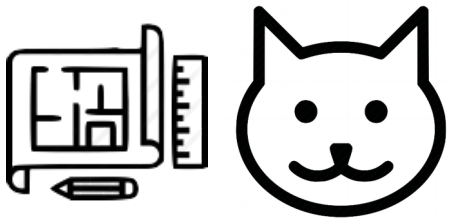
contribution I

Scalable smart contract platform

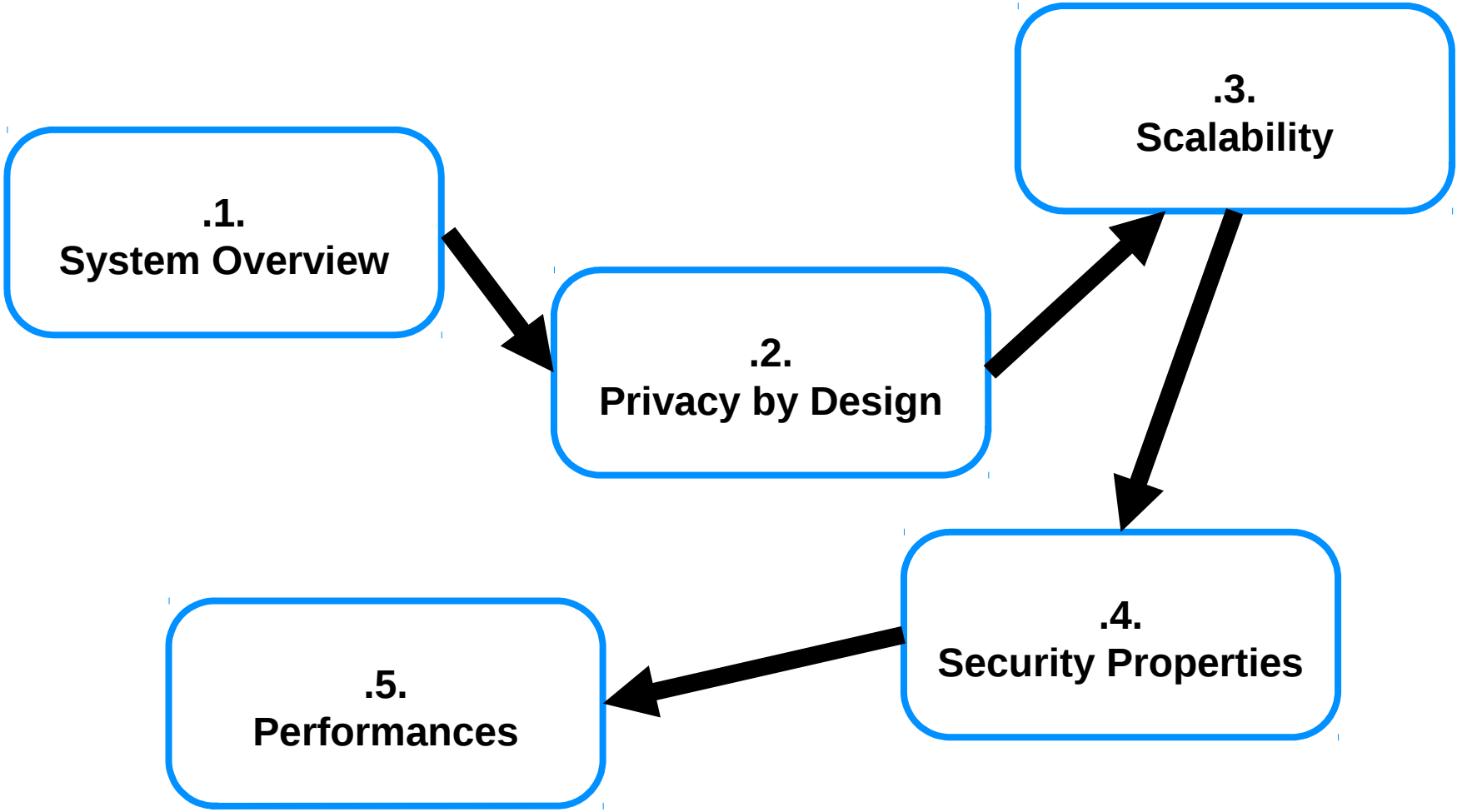


contribution II

Supporting PETs by Design

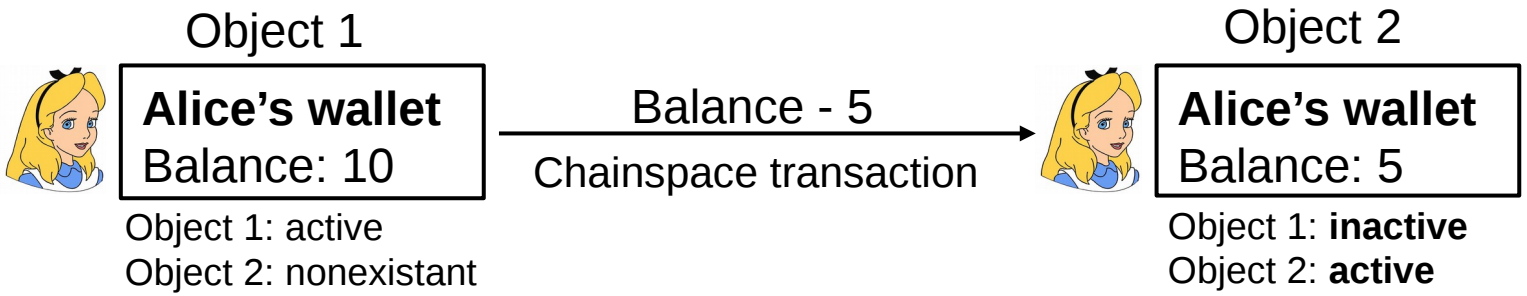


Contents


.1.
System Overview
.2.
Privacy by Design
.3.
Scalability
.4.
Security Properties
.5.
Performances

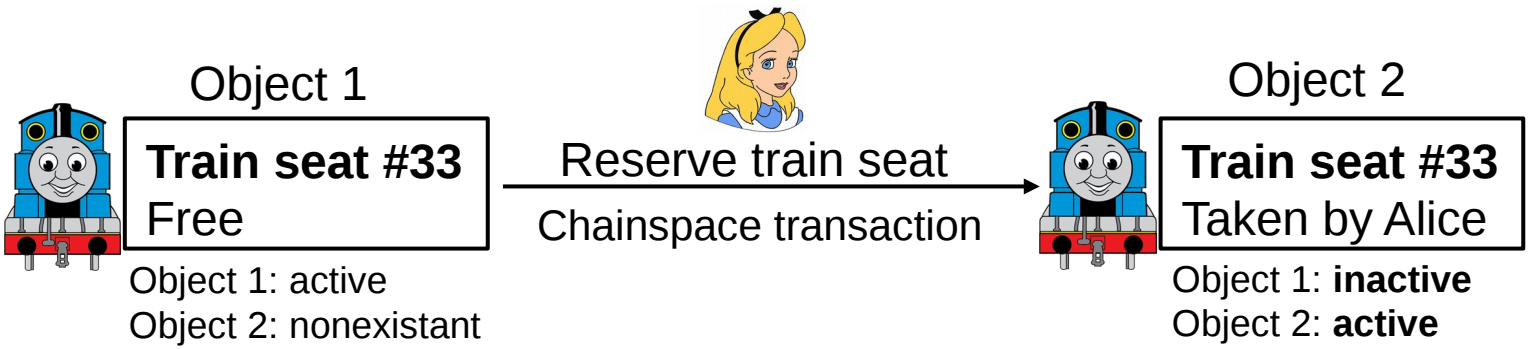
System Overview

- How does Chainspace work?
  - Everything in Chainspace is an **object**
  - For example: a bank account, a hotel room, a train seat
  - Objects are either **active**, **inactive**, or **nonexistant**
  - Only **active** objects can be used in transactions



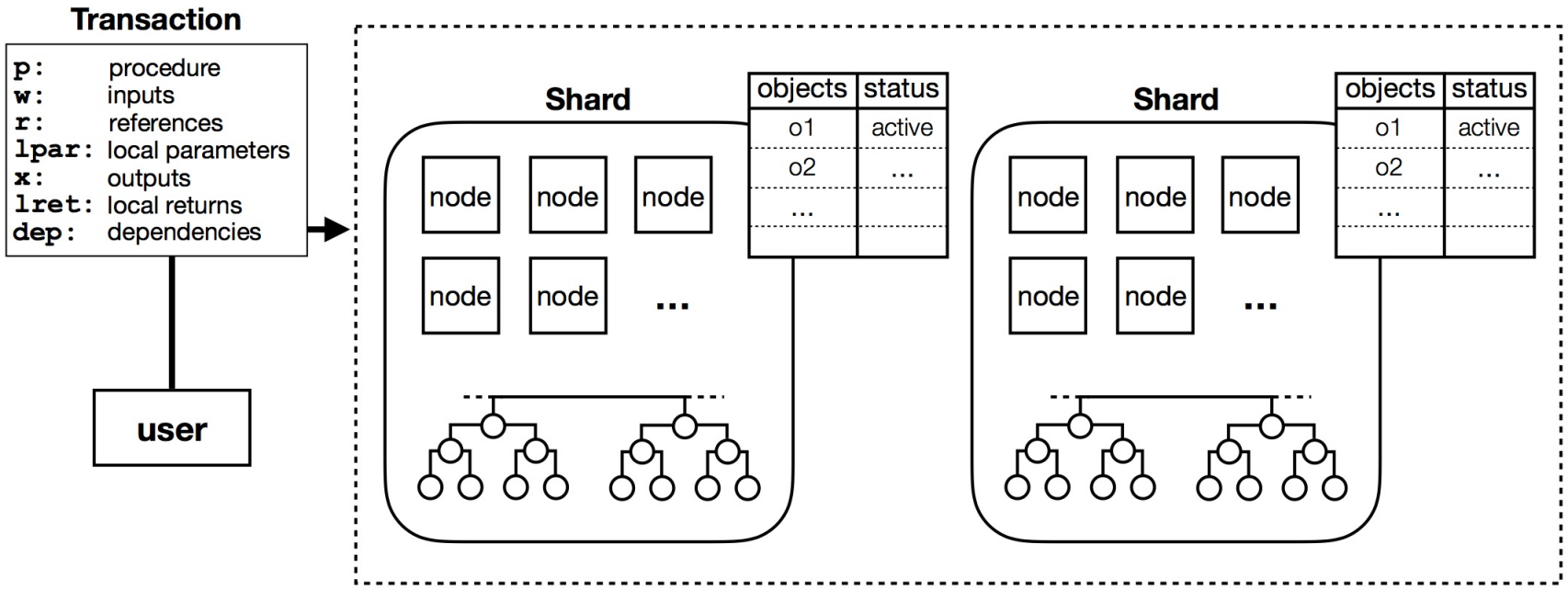
System Overview

- How does Chainspace work?
  - Everything in Chainspace is an **object**
  - For example: a bank account, a hotel room, a train seat
  - Objects are either **active**, **inactive**, or **nonexistant**
  - Only **active** objects can be used in transactions

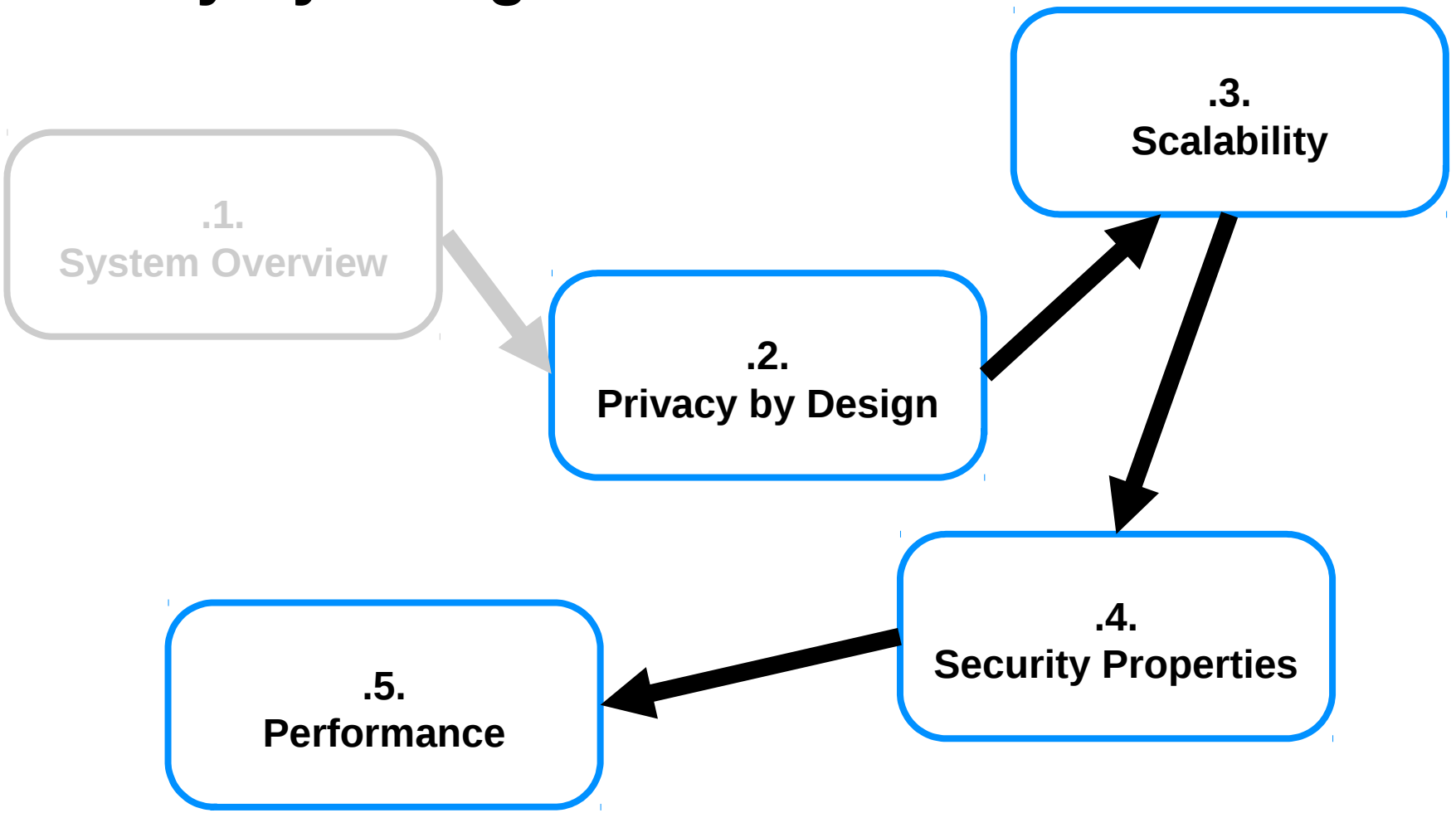


System Overview

- How does Chainspace work?
  - Nodes are organised into **shards**
  - Shards manage **objects**
  - Objects can be used only once

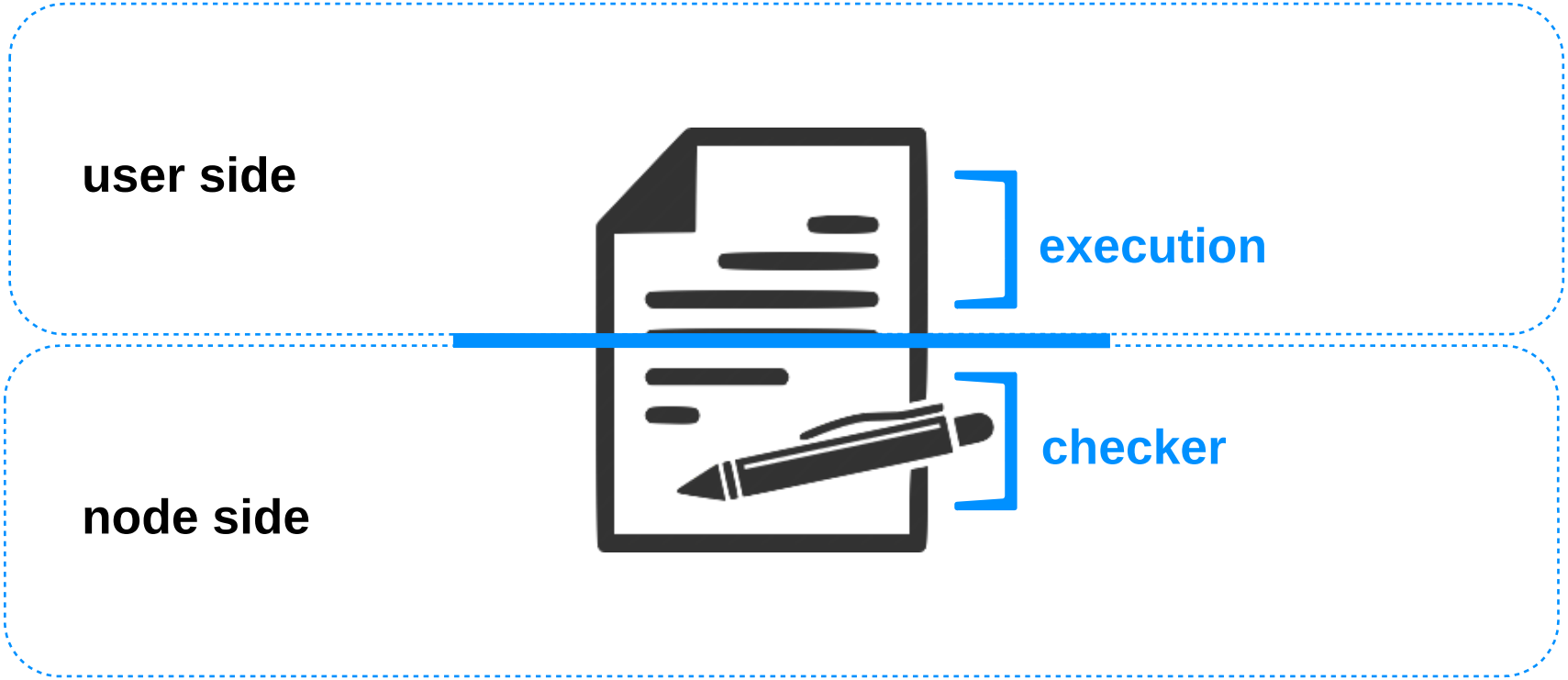


Privacy by Design



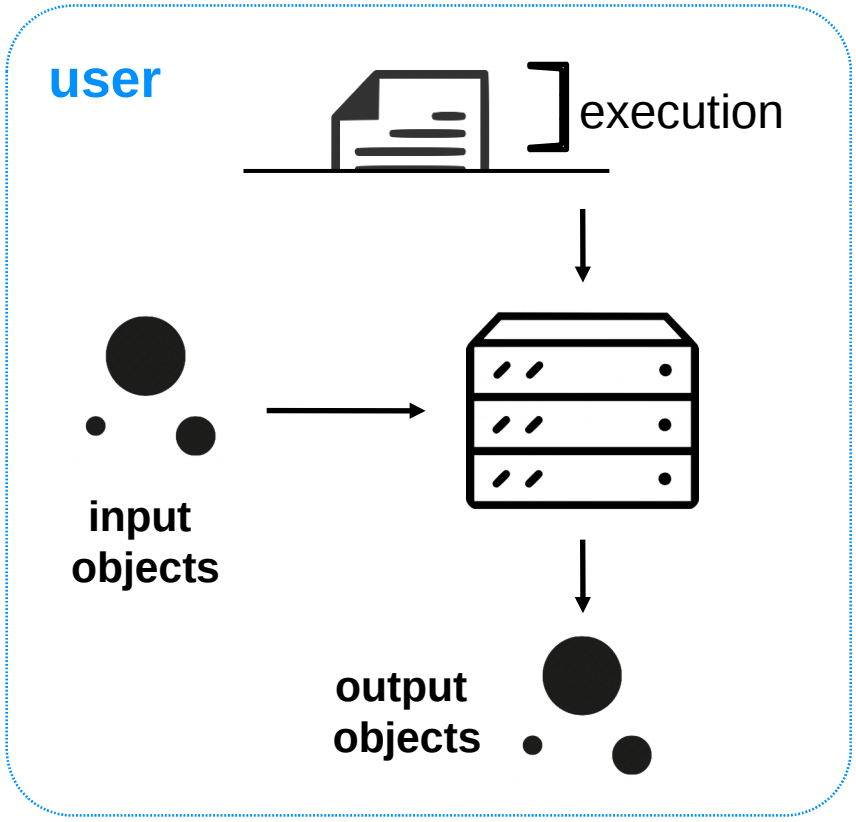
Privacy by Design

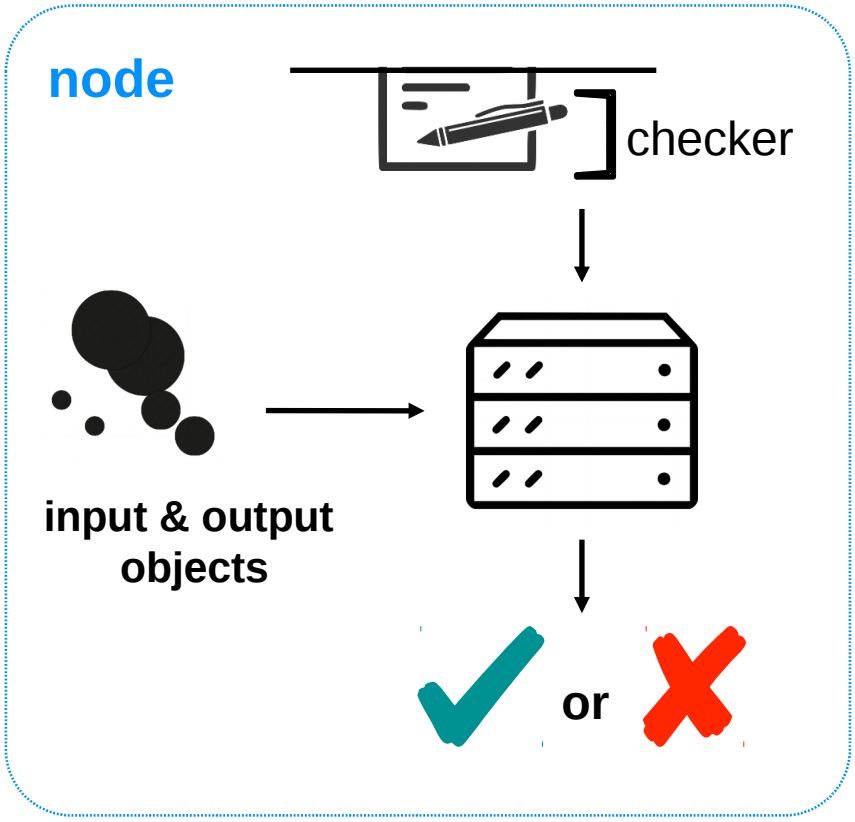
- What are Chainspace Smart Contract?



Privacy by Design

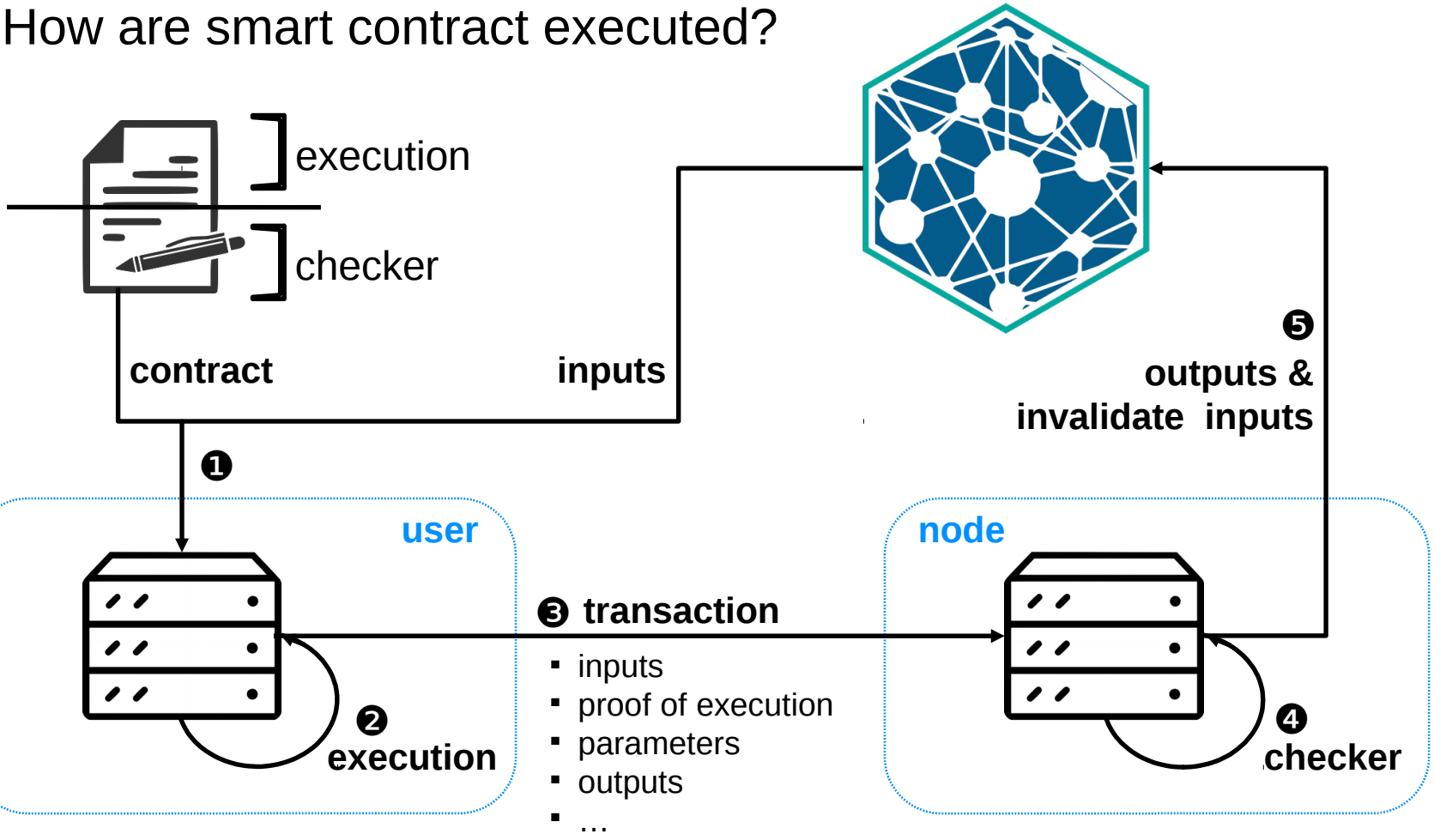
- What are Chainspace Smart Contract?





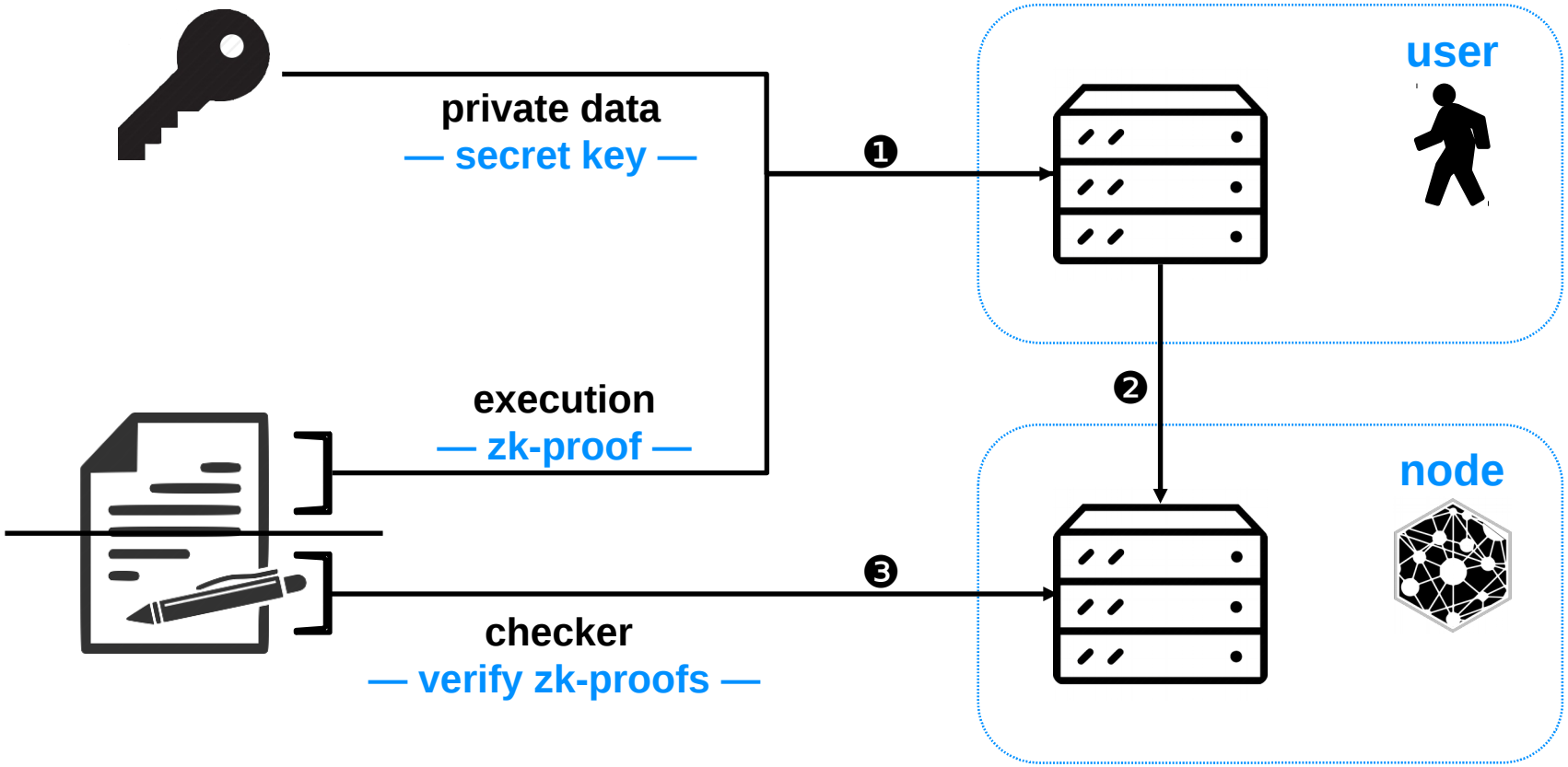
Privacy by Design

- How are smart contract executed?



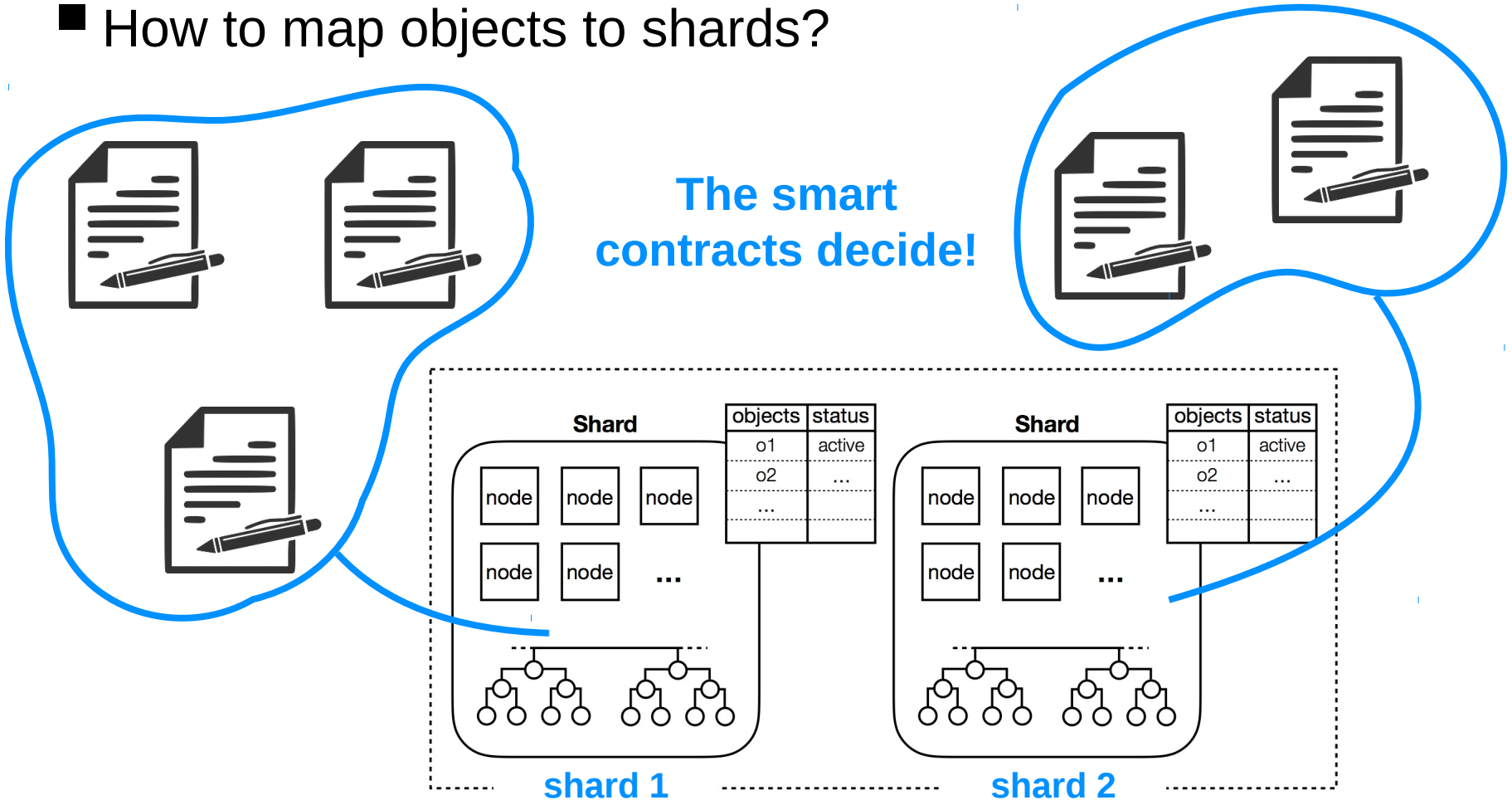
Privacy by Design

- Private data never leaves the client!

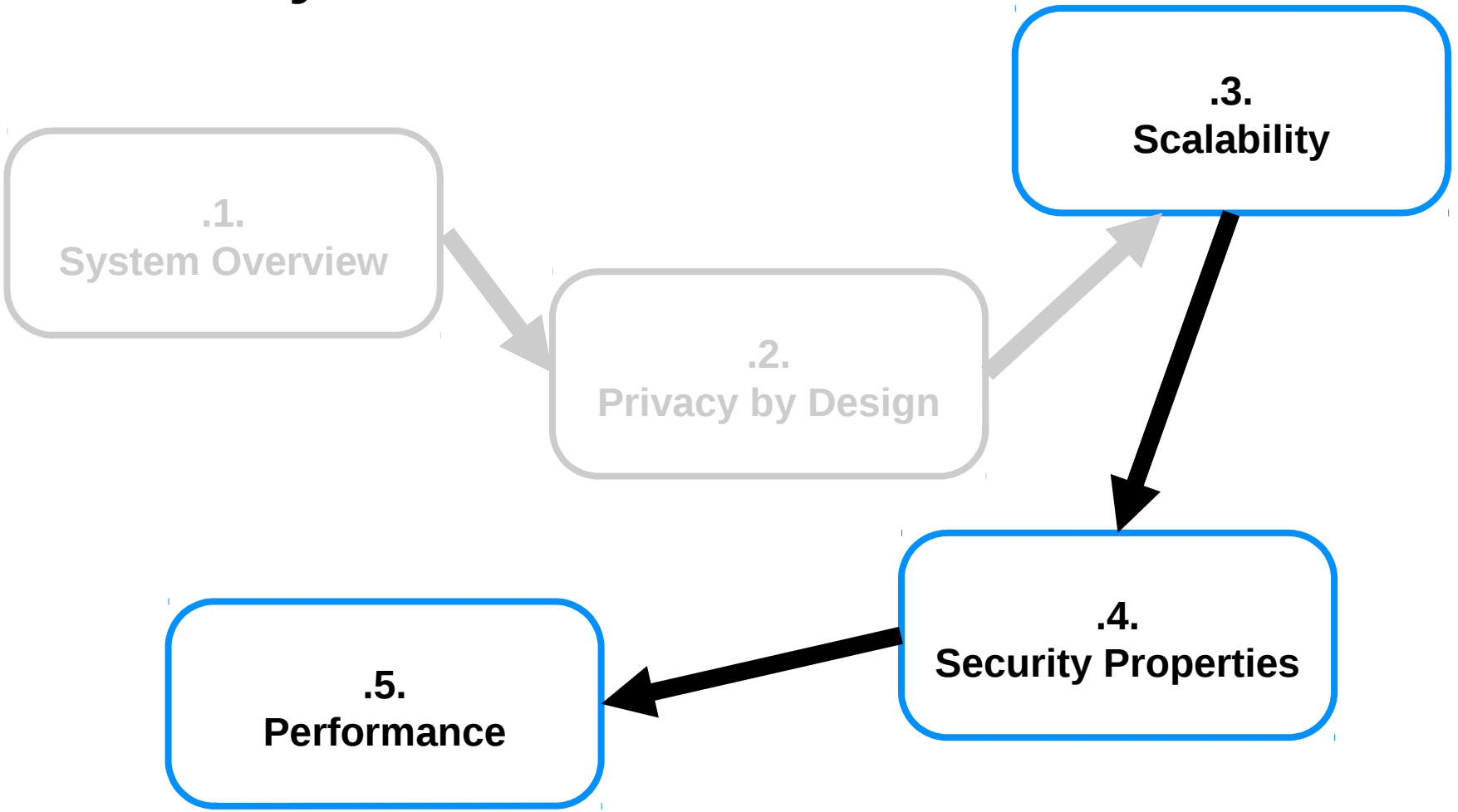


Privacy by Design

- How to map objects to shards?

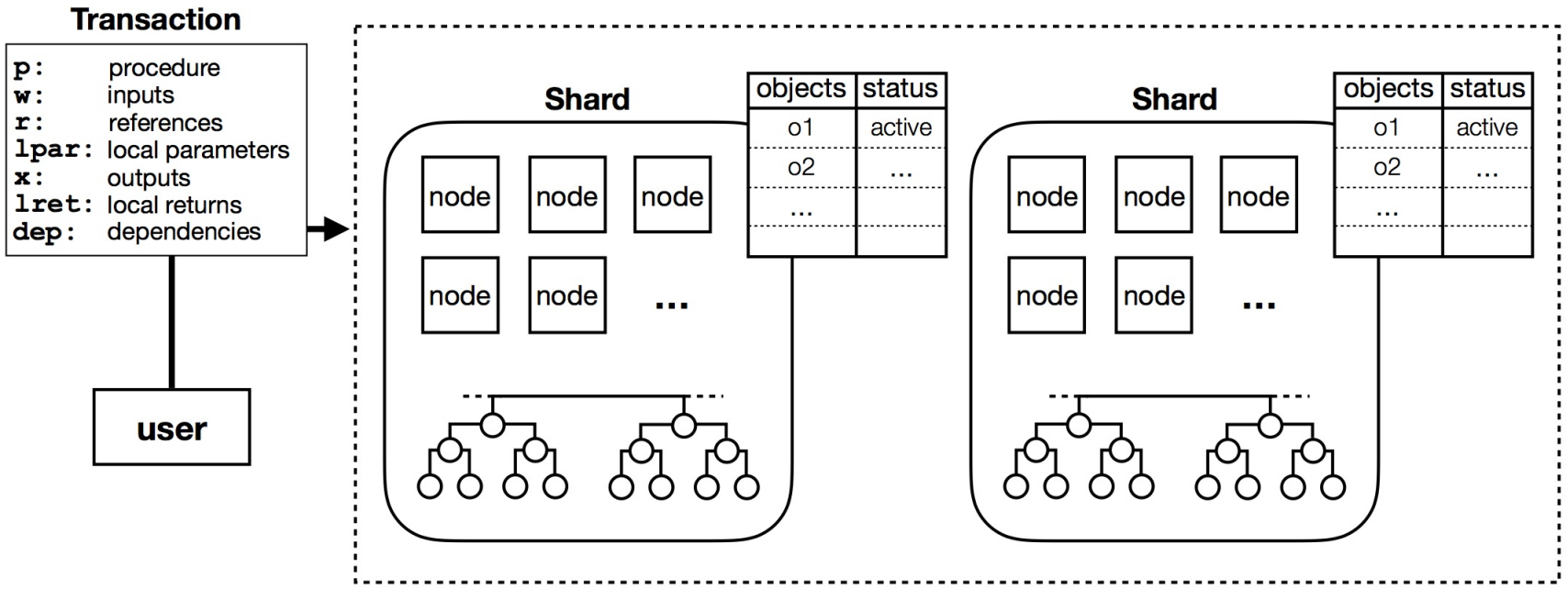


Scalability



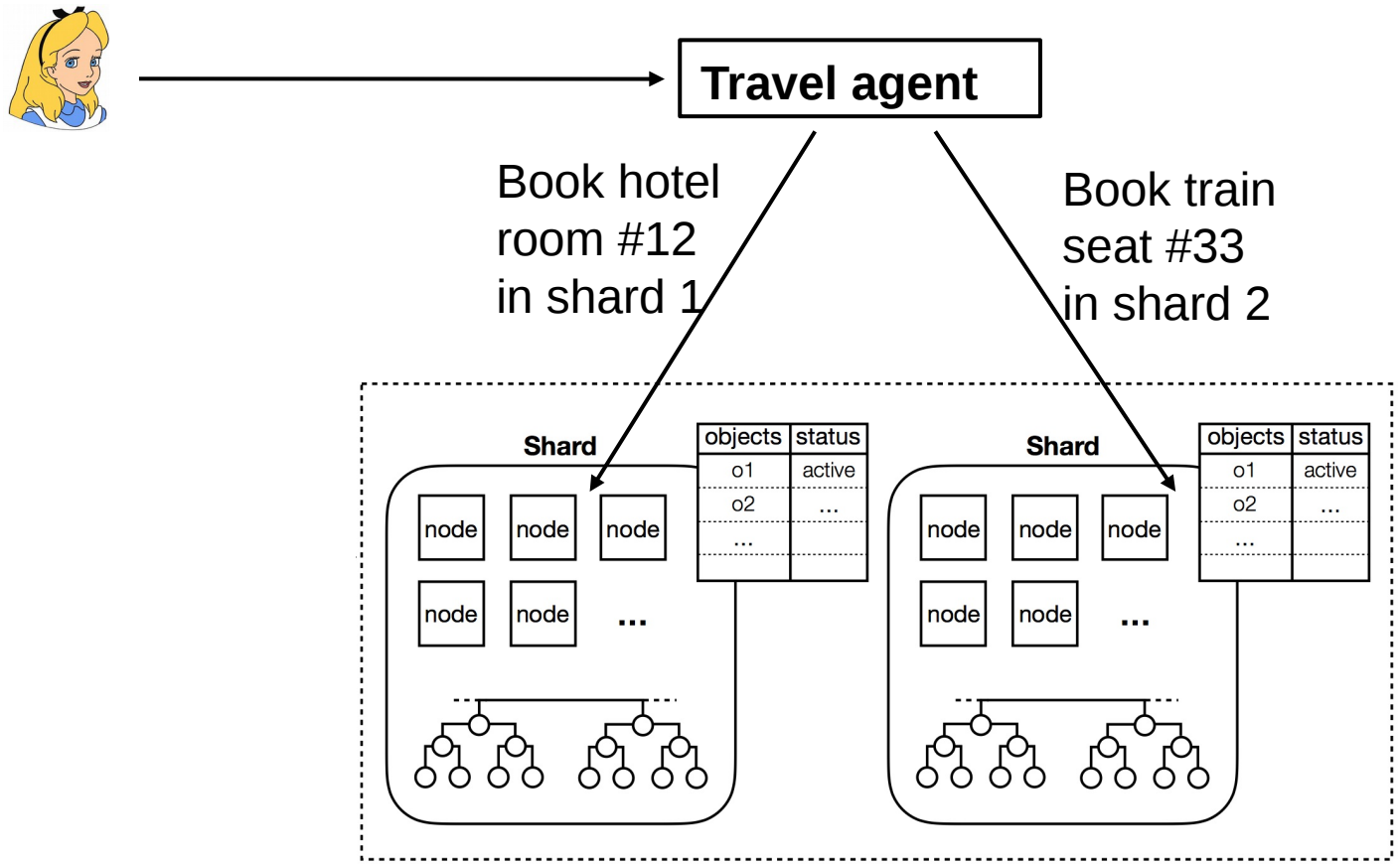
Scalability

- We split the blockchain to multiple shards.



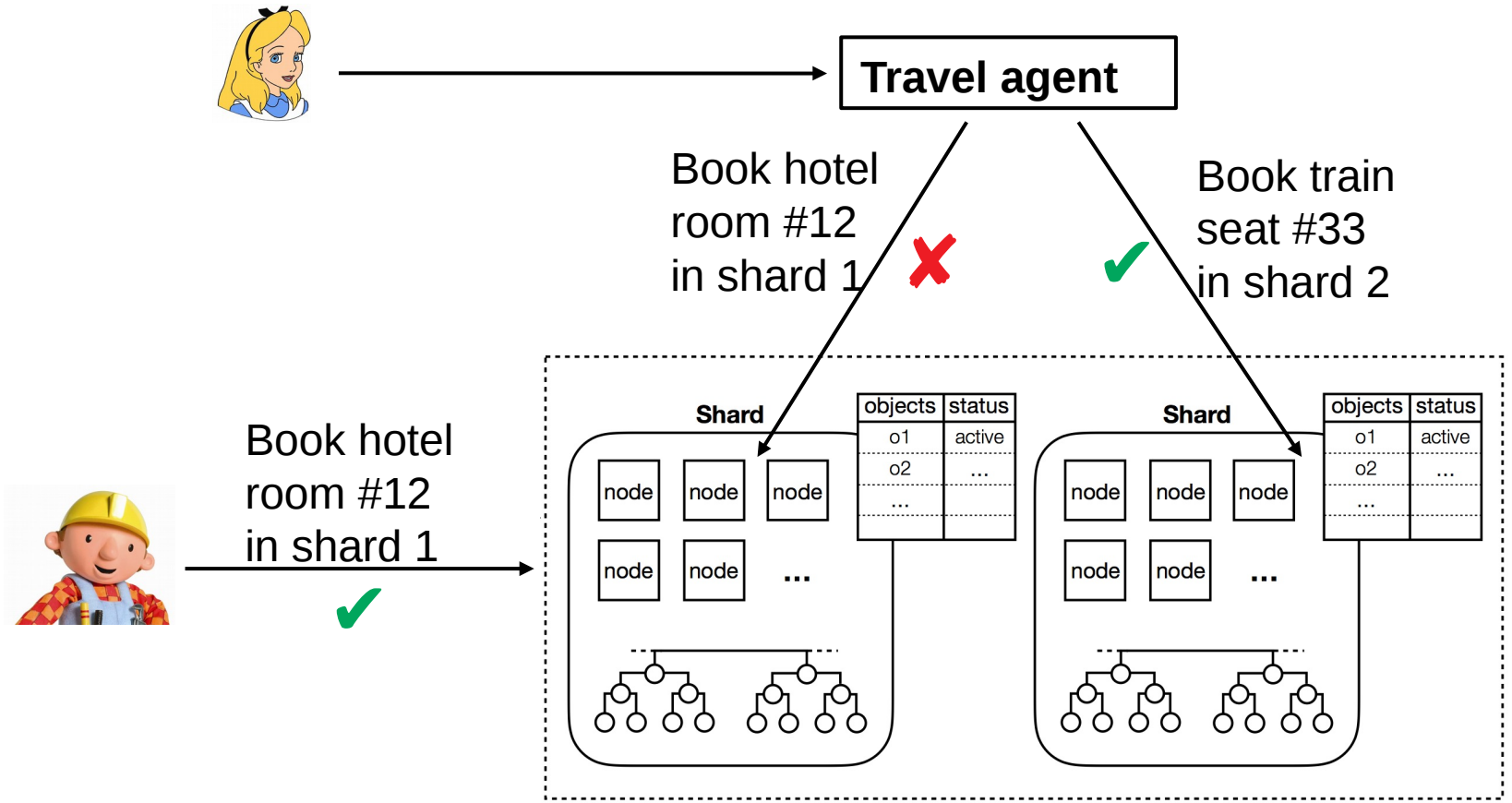
Scalability

- What is the train-and-hotel problem?



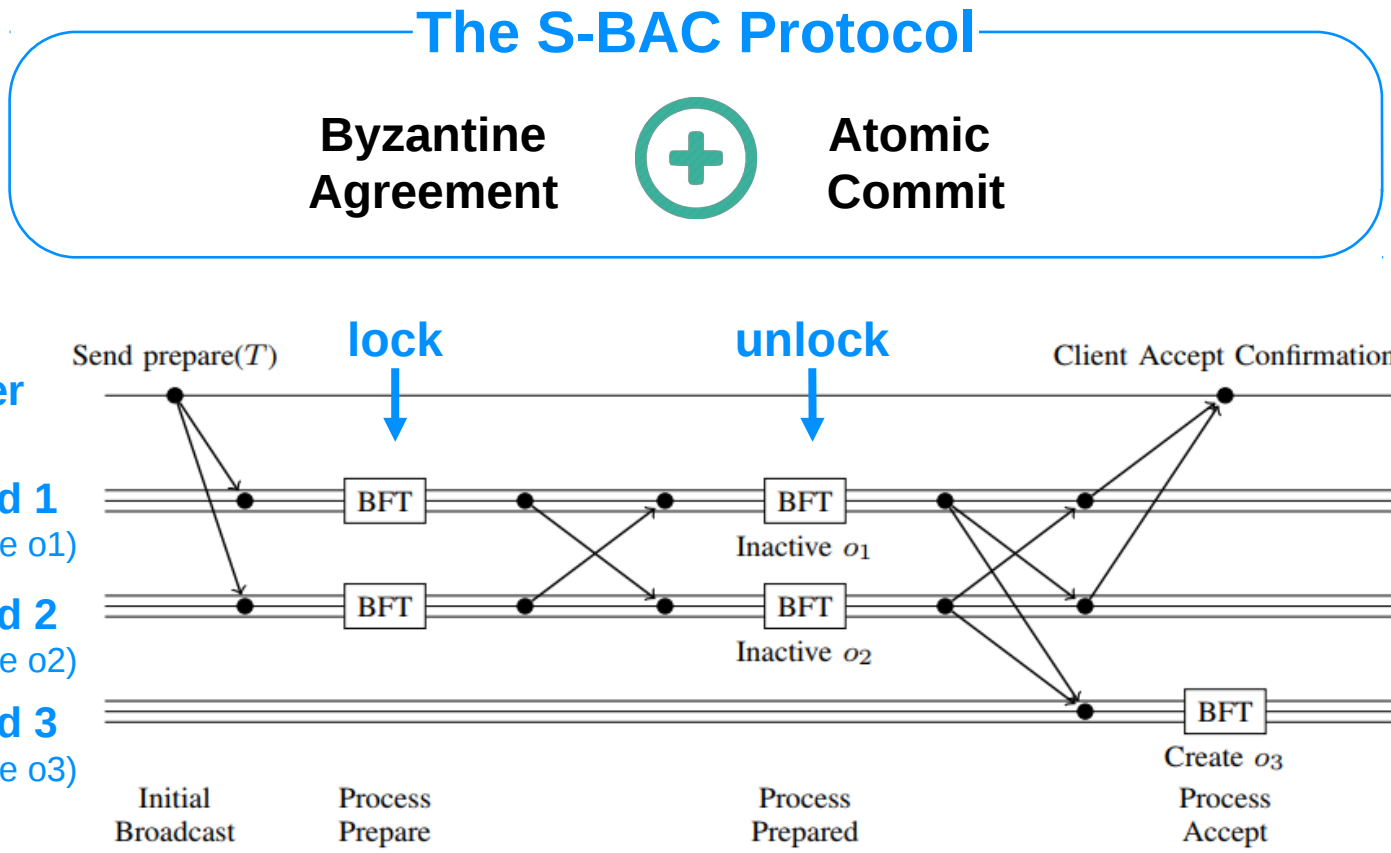
Scalability

- What is the train-and-hotel problem?



Scalability

- How nodes reach consensus?

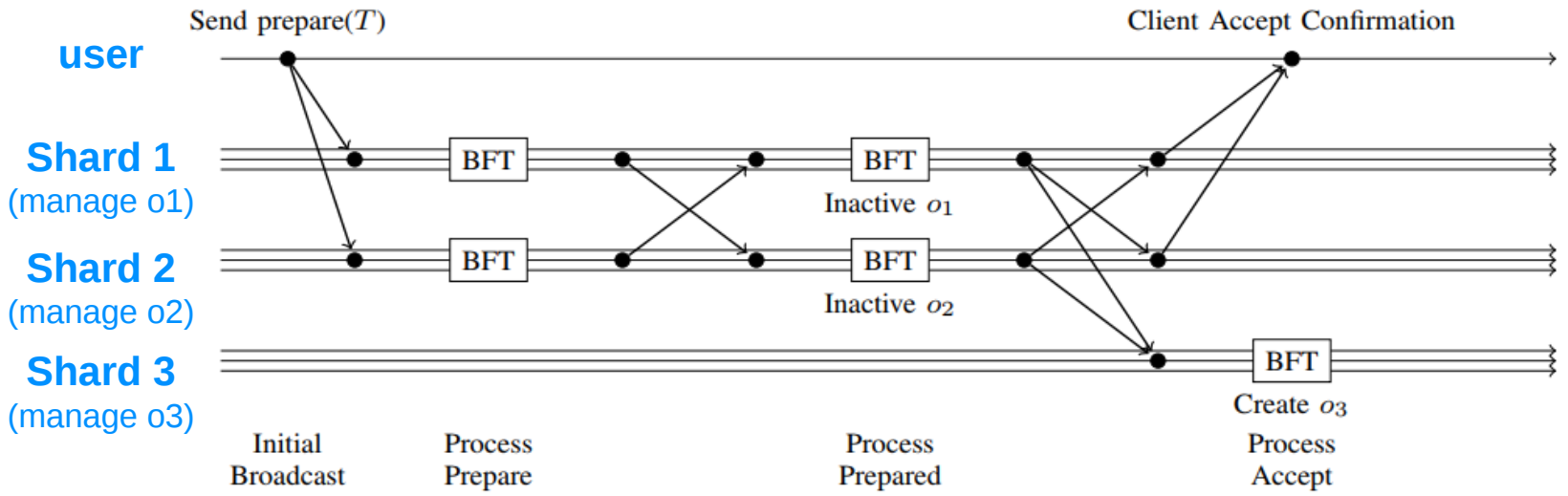


Scalability

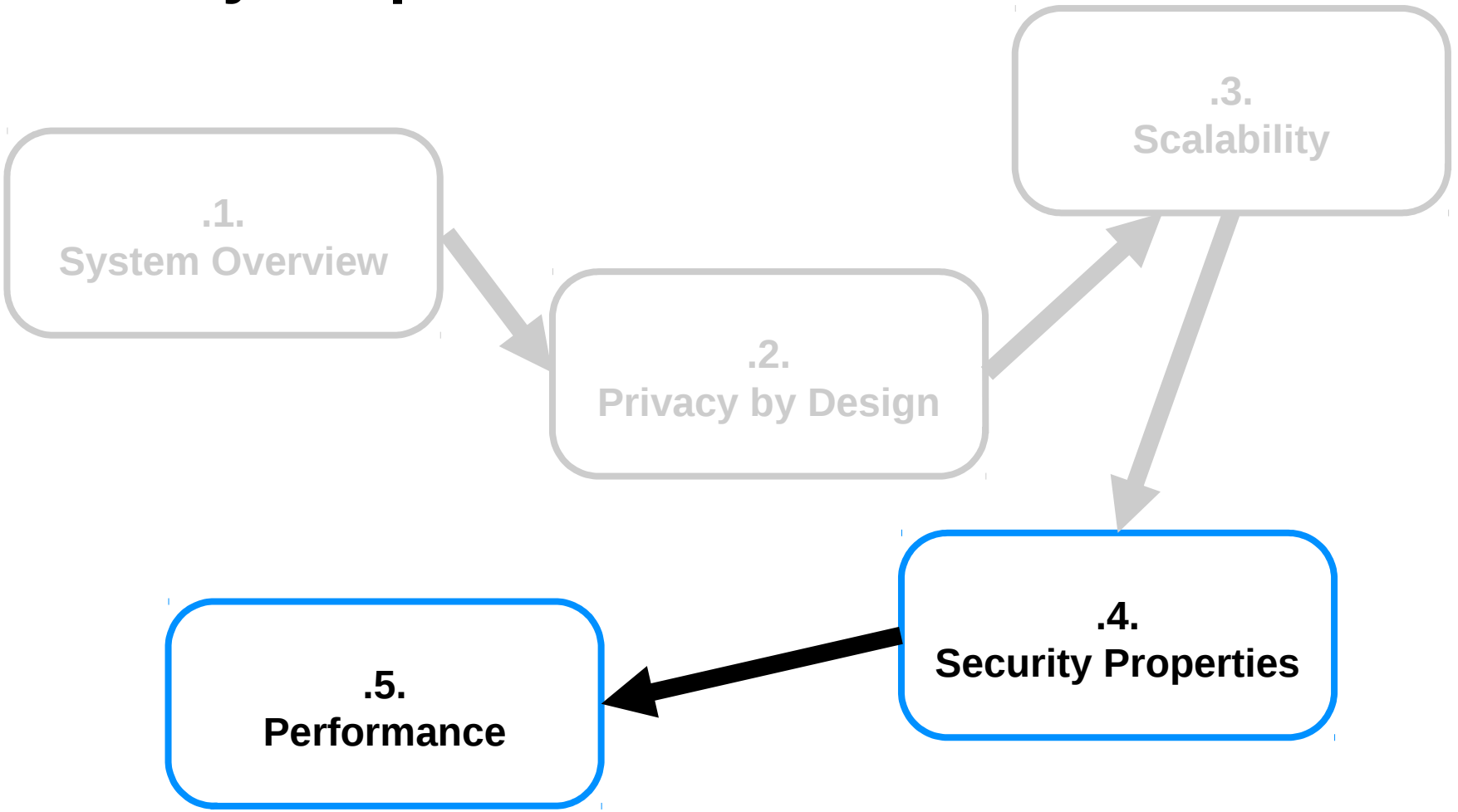
- The Wisdom behind S-BAC

Only shards managing *o1* and *o2* are working

Shard 1 and shard 2 can work in parallel



Security Properties



Security Properties

- What does Chainspace guarantee?
  - **Honest Shard (HS):** among ***3f+1*** nodes, at most ***f*** are malicious.
  - **Malicious Shard (DS):** over ***f*** dishonest nodes.
  - Chainspace properties:

Transparency (HS & DS)

Anyone can authenticate the history of transactions and objects that led to the creation of an object.

Encapsulation (HS & DS)

A smart contract cannot interfere with objects created by another contract (except if defined by that contract).

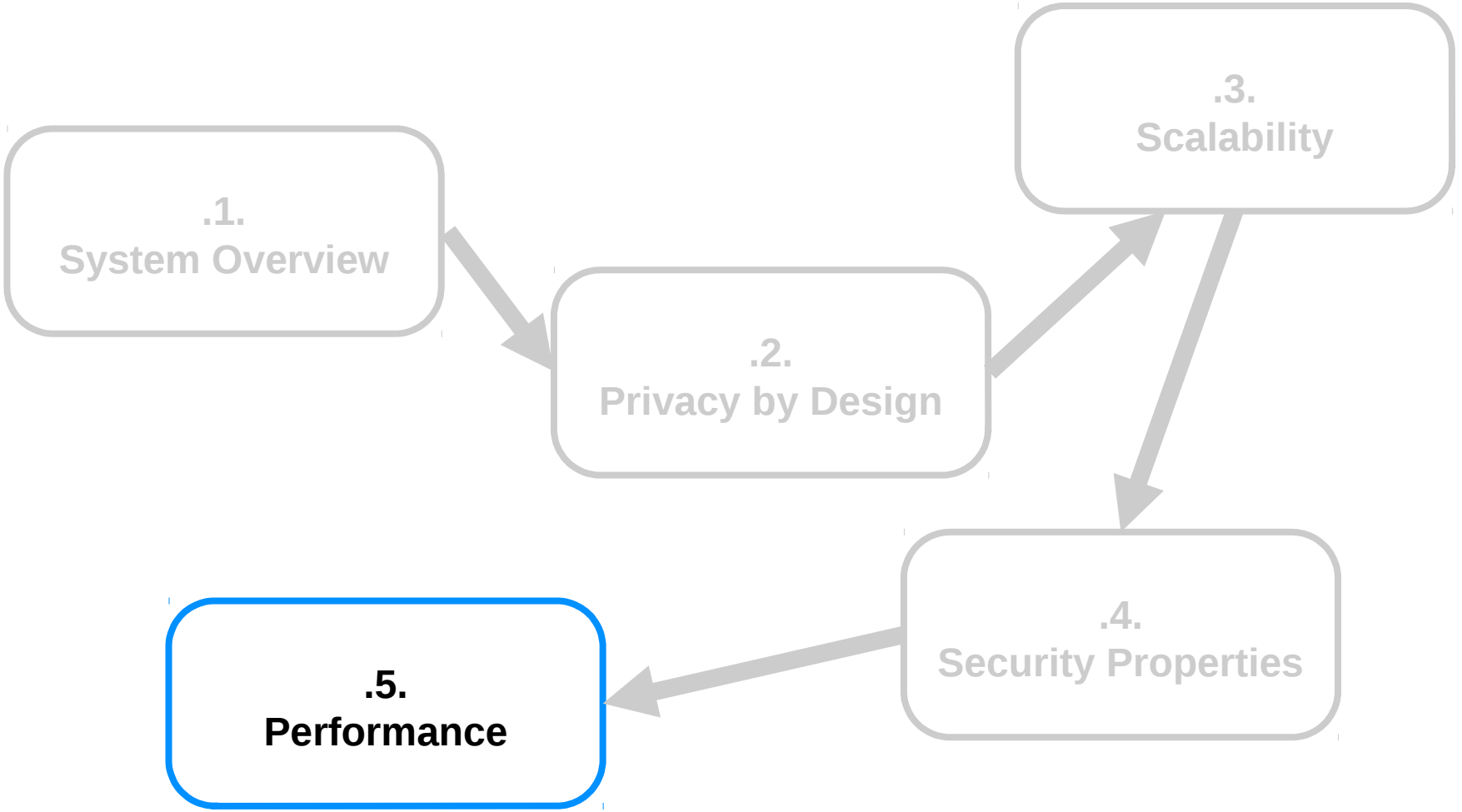
Integrity (HS)

Only valid & non-conflicting transactions will be executed.

Non-Repudiation (HS & DS)

Misbehaviour is detectable: there are evidences of misbehaviour pointing to the faulty parties or shards.

Performance



Performance

- What did we implemented?

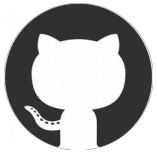
Deployed and tested on Amazon AWS

S-BAC protocol implemented in Java

Based on BFT-SMaRt

Everything is released as open source software

`https://github.com/chainspace`

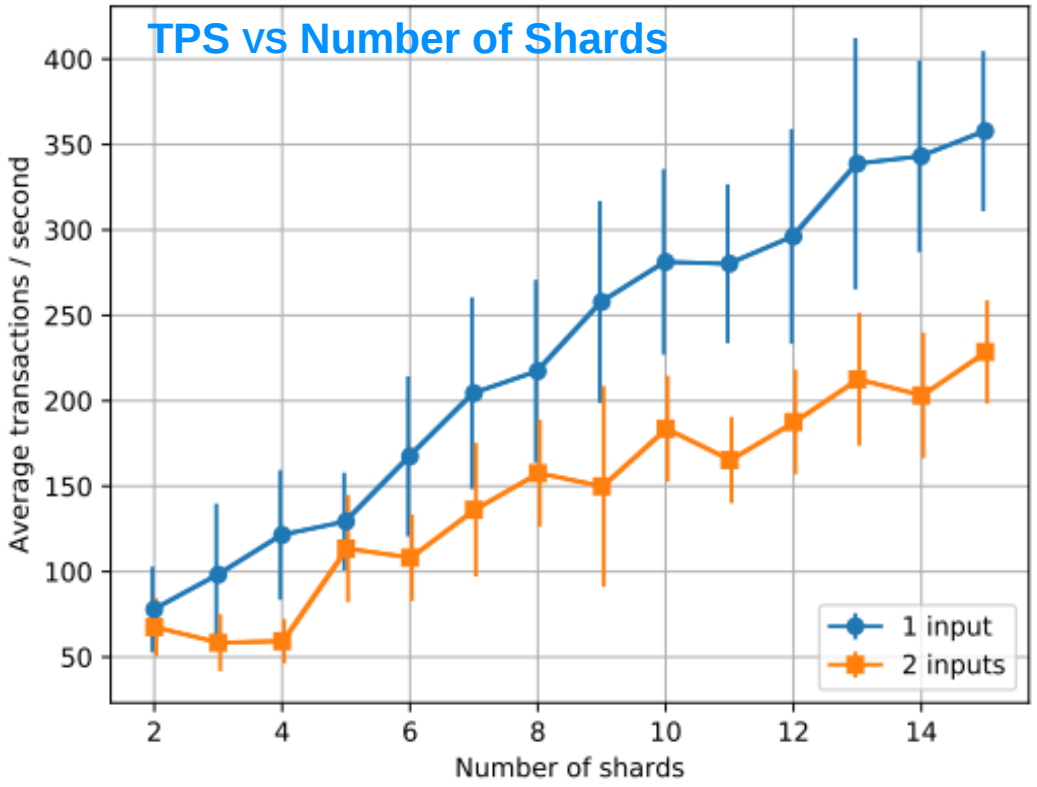


Python contract environment

1. Helps developers
2. Simulation of the checker
3. No need for full deployment

Performance

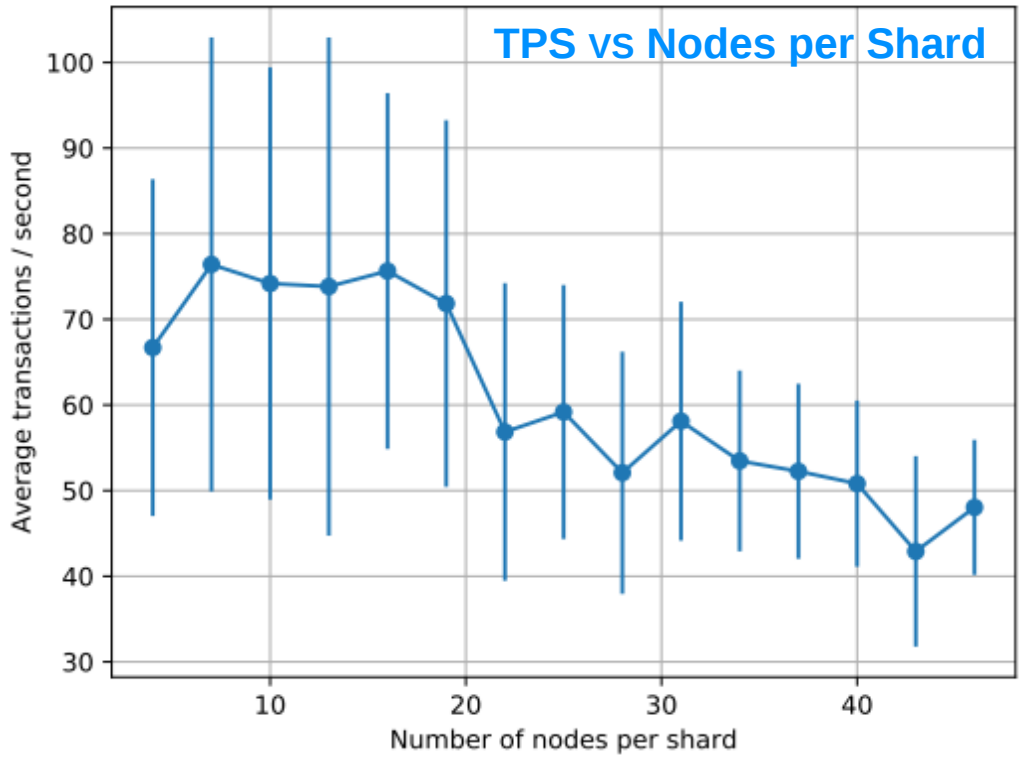
- How the number of shards influence the TPS?



(standard setup: 2 shards, 4 nodes/shard, 20 runs/data point)

Performance

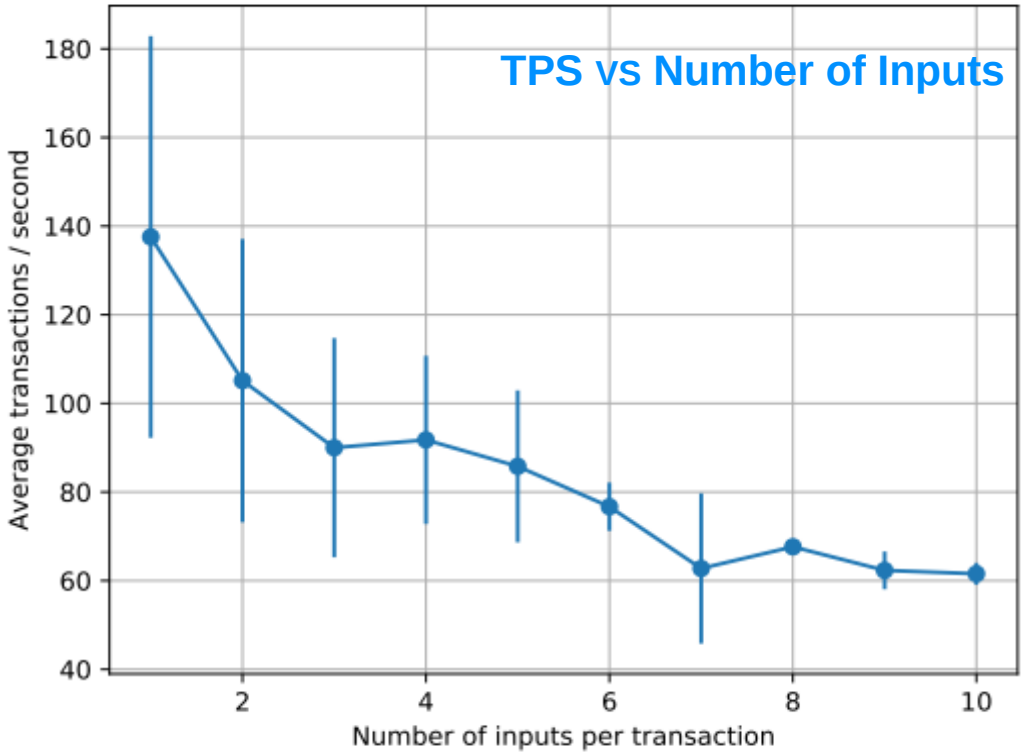
- How does the size of the shard influence the TPS?



(standard setup: 2 shards, 4 nodes/shard, 20 runs/data point)

Performance

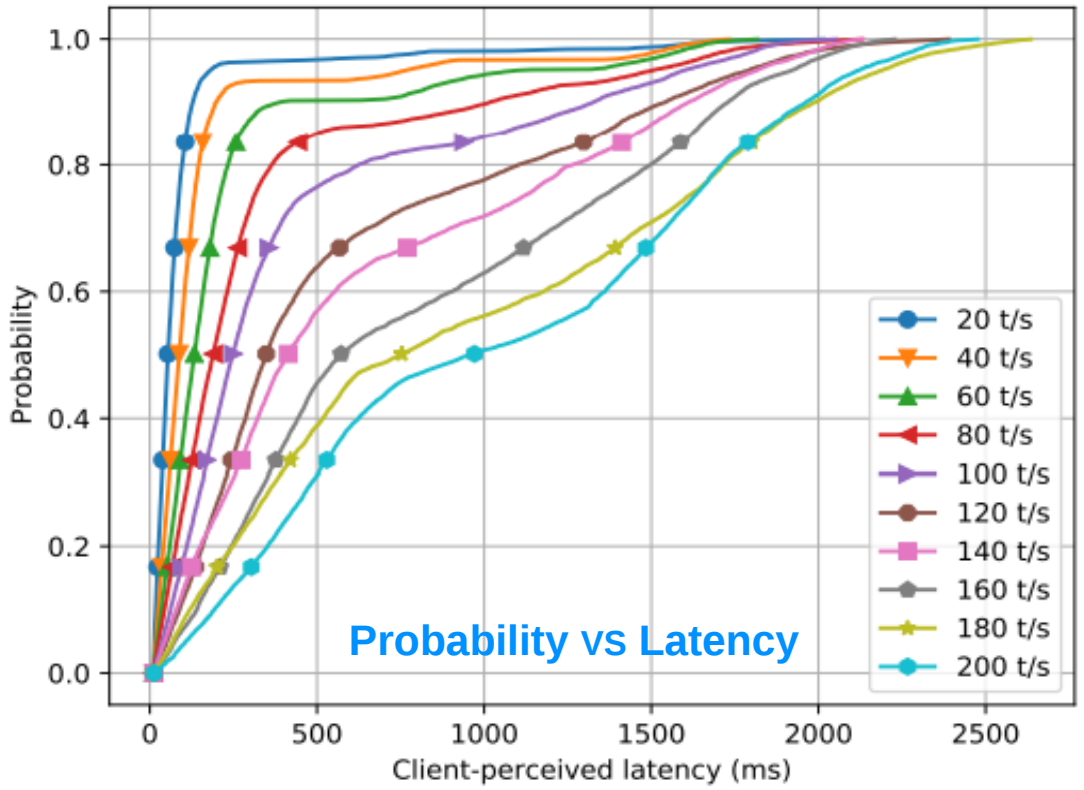
- How the number of inputs influence the TPS?



(standard setup: 2 shards, 4 nodes/shard, 20 runs/data point)

Performance

- How does the latency vary under different system loads?



(standard setup: 2 shards, 4 nodes/shard, 20 runs/data point)

Conclusions

- What else is in the paper?

Cross shard transactions

Real world applications (smart metering, …)

Smart contracts benchmarking

And much more…

Chainspace: A Sharded Smart Contracts Platform

Mustafa Al-Bassam*, Alberto Sonnino*, Shehar Bano*, Dave Hrycyszyn† and George Danezis*
* University College London, United Kingdom
† constructiveproof.com

***Abstract*—Chainspace is a decentralized infrastructure, known as a distributed ledger, that supports user defined smart contracts and executes user-supplied transactions on their objects. The correct execution of smart contract transactions is verifiable by all. The system is scalable, by sharding state and the execution of transactions, and using *S*-BAC, a distributed commit protocol, to guarantee consistency. Chainspace is secure against subsets of nodes trying to compromise its integrity or availability properties through Byzantine Fault Tolerance (BFT), and extremely high-auditability, non-repudiation and 'blockchain' techniques. Even when BFT fails, auditing mechanisms are in place to trace malicious participants. We present the design, rationale, and details of Chainspace; we argue through evaluating an implementation of the system about its scaling and other features; we illustrate a number of privacy-friendly smart contracts for smart metering, polling and banking and measure their performance.**

I. Introduction

Chainspace is a distributed ledger platform for high-integrity and transparent processing of transactions within a decentralized system. Unlike application specific distributed ledgers, such as Bitcoin [Nak08] for a currency, or certificate transparency [LLK13] for certificate verification, Chainspace offers extensibility though smart contracts, like Ethereum [Woo14]. However, users expose to Chainspace enough information about contracts and transaction semantics, to provide higher scalability through sharding across infrastructure nodes: our modest testbed of 60 cores achieves 350 transactions per second, as compared with a peak rate of less than 7 transactions per second for Bitcoin over 6K full nodes. Etherium currently processes 4 transactions per second, out of theoretical maximum of 25. Furthermore, our platform is agnostic as to the smart contract language, or identity infrastructure, and supports privacy features through modern zero-knowledge techniques (BCCG16, DGFK14).

Unlike other scalable but 'permissioned' smart contract platforms, such as Hyperledger Fabric [Cac16] or BigchainDB [MMM+16], Chainspace aims to be an 'open' system: it allows anyone to author a smart contract, anyone to provide infrastructure on which smart contract code and state runs, and any user to access calls to smart contracts. Further, it provides ecosystem features, by allowing composition of smart contracts from different authors. We integrate a value system, named CSCoin, as a system smart contract to allow for accounting between those parties.

However, the security model of Chainspace, is different from traditional unpermissioned blockchains, that rely on proof-of-work and global replication of state, such as Ethereum. In Chainspace smart contract authors designate the parts of the infrastructure that are trusted to maintain the integrity of their contract—and only depend on their correctness, as well as the correctness of contract sub-calls. This provides fine grained control of which part of the infrastructure need to be trusted on a per-contract basis, and also allows for horizontal scalability.

This paper makes the following contributions:

- It presents Chainspace, a system that can scale arbitrarily as the number of nodes increase, tolerates byzantine failures, and can be fully and publicly audited.
- It presents a novel distributed atomic commit protocol, called *S*-BAC, for sharding generic smart contract transactions across multiple byzantine nodes, and correctly coordinating those nodes to ensure safety, liveness and security properties.
- It introduces a distinction between parts of the smart contract that execute a computation, and those that check the computation and discusses how that distinction is key to supporting privacy-friendly smart-contracts.
- It provides a full implementation and evaluates the performance of the byzantine distributed commit protocol, *S*-BAC, on a real distributed set of nodes and under varying transaction loads.
- It presents a number of key system and application smart contracts and evaluates their performance. The contracts for privacy-friendly smart-metering and privacy-friendly polls illustrate and validate support for high-integrity and high-privacy applications.

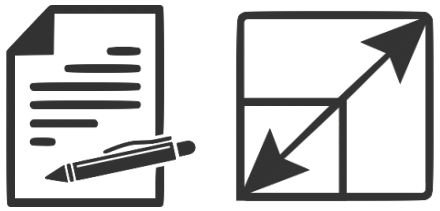
Outline: Section II presents an overview of Chainspace; Section III presents the client-facing application interface; Section IV presents the design of internal data structures guaranteeing integrity, the distributed architecture, the byzantine commit protocols, and smart contract definition and composition. Section V argues the correctness and security; specific smart contracts and their evaluations are presented in Section VI; Section VII presents an evaluation of the core protocols and smart contract performance; Section VIII presents limitation and Section IX a comparison with related work; and Section X concludes.

Conclusions

- What did we talk about?

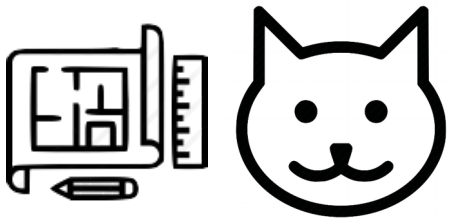
contribution I

Scalable smart contract platform



contribution II

Supporting PETs by Design



Conclusions

- **Main take-aways**

sharding  **scalability**

execution / checker  **privacy by design**

Conclusions

- Future Works

1. How to recover from malicious shards?

2. How can a smart contract creator avoid dishonest shards?

Conclusions

- Future Works

3. How to bootstrap the system?

4. How to incentivise nodes?

Thank you for your attention

Questions?

Mustafa Al-Bassam
m.albassam@cs.ucl.ac.uk