Chainspace:
A Sharded Smart Contract Platform

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

* University College London
** constructiveproof.com

NDSS Symposium 2018

The authors

Mustafa Al-Bassam

Alberto Sonnino

Bano Shehar

Dave Hrycyszyn

George Danezis

People love blockchains

- ✓ Fancy
- ✓ Involve money
- ✓ Look complicated
- ✓ About security stuff

But what are blockchains?

But what are blockchains?

- In a few words (simplified):

Systems to store records that can be verified by anyone, that no-one can modify, and without a central authority

But what are blockchains?

- In a few words (simplified):

Systems to store records that can be verified by anyone, that no-one can modify, and without a central authority

Publicly verifiable

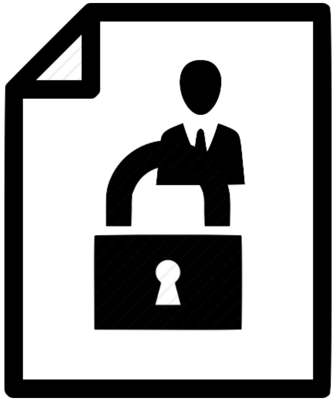
But what are blockchains?

- In a few words (simplified):

Systems to store records that can be verified by anyone, that no-one can modify, and without a central authority

Publicly verifiable

Immutable

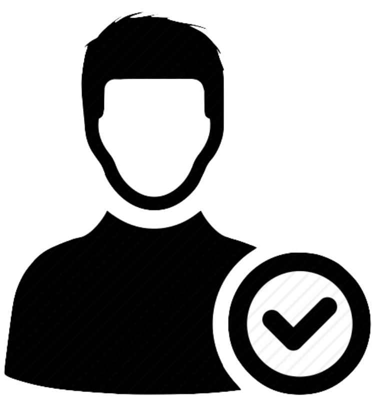


But what are blockchains?

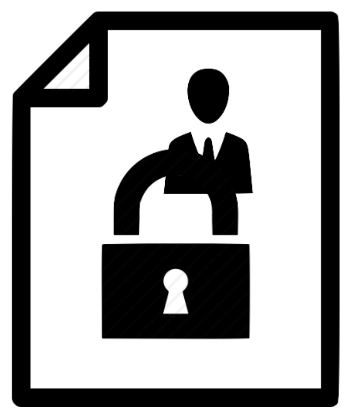
- In a few words (simplified):

Systems to store records that can be verified by anyone, that no-one can modify, and without a central authority

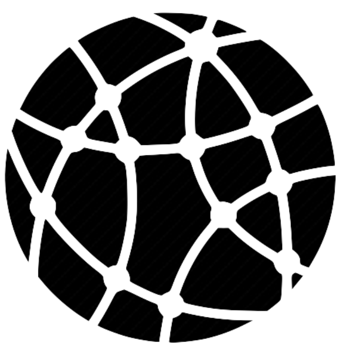
Publicly verifiable



Immutable



Decentralised



What are smart contracts?

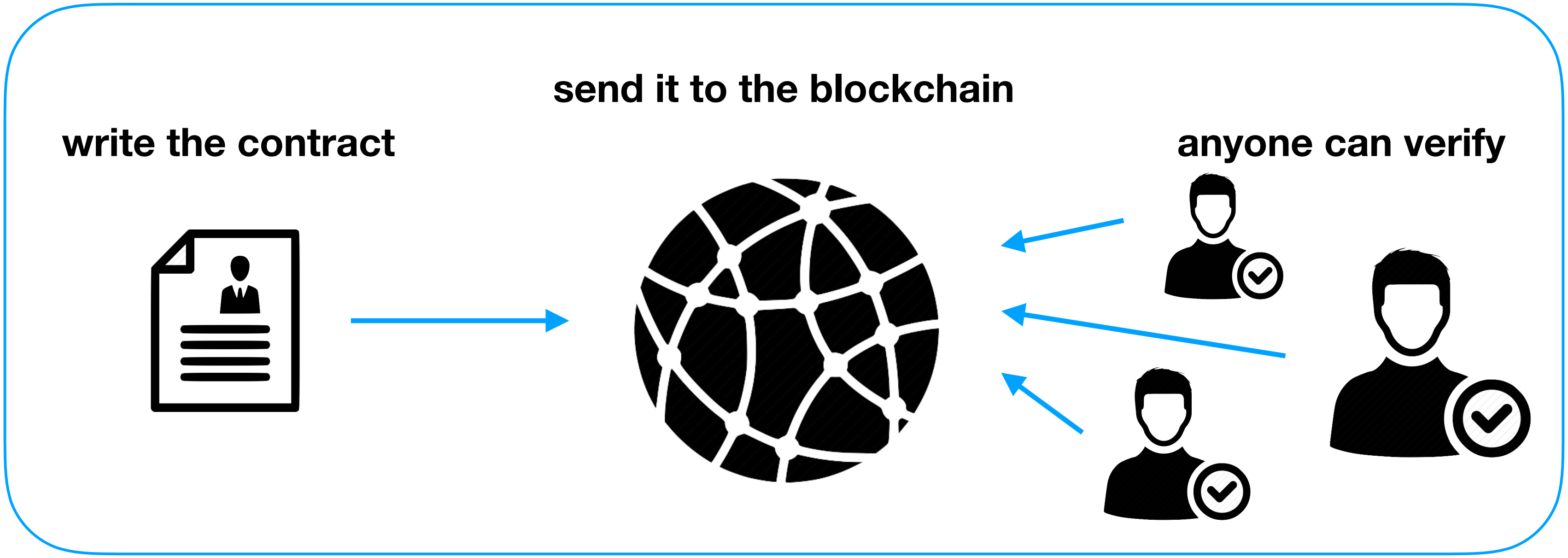
- In a few words (simplified):

Smart contracts are computer programs that are 'executed' on the blockchain

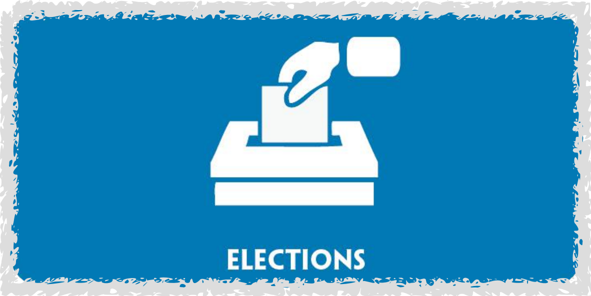
What are smart contracts?

- In a few words (simplified):

Smart contracts are computer programs that are 'executed' on the blockchain



What can we do with that?



? ? ?

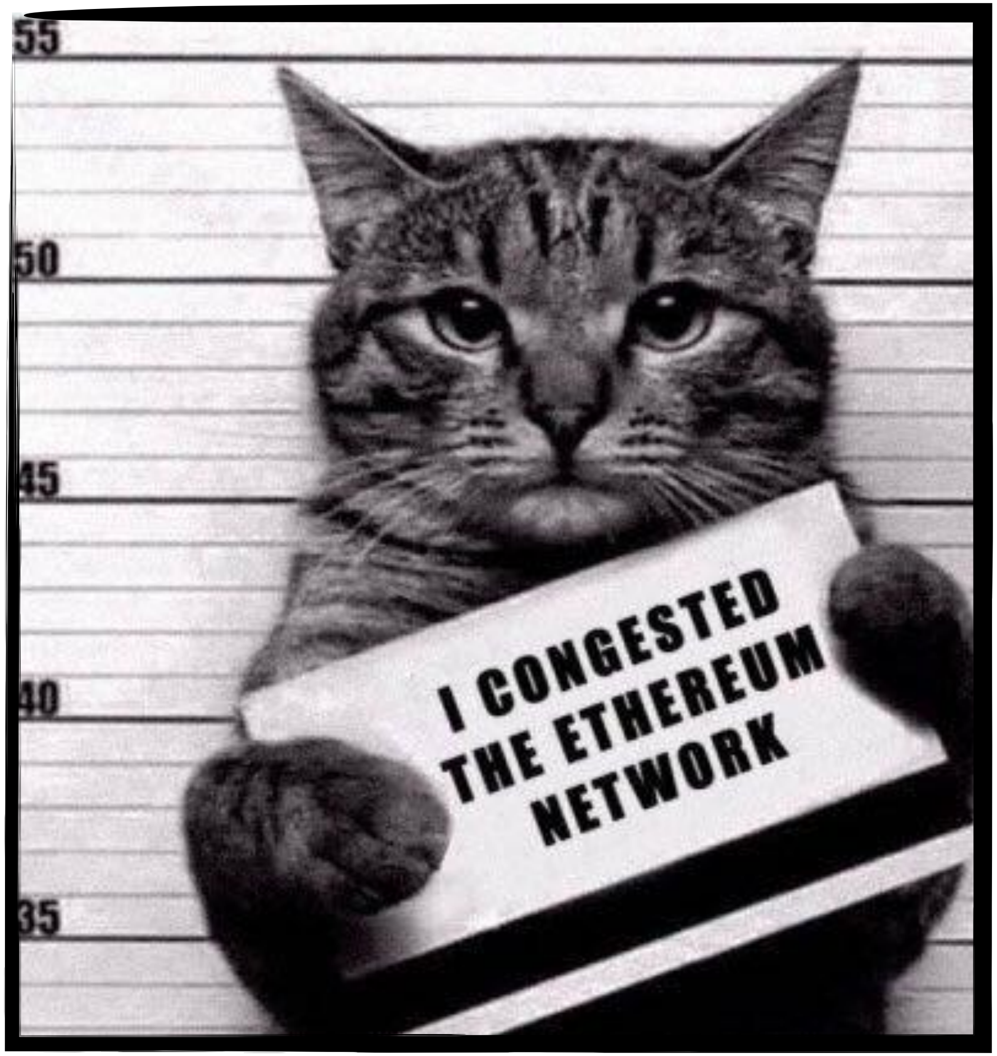
or…

When blockchains meet cats…

When blockchains meet cats…

Crypto-kitties

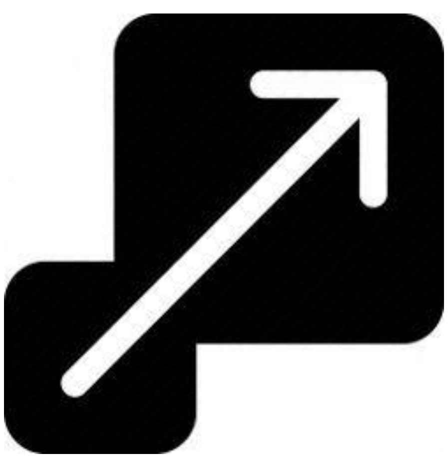
When blockchains meet kittens…

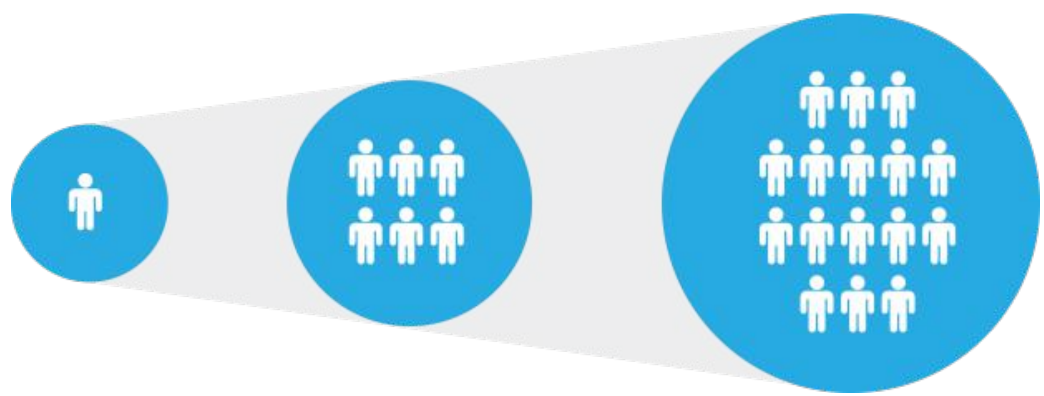


Why did that happen?

- **Blockchains do not scale!**

No matter how many computer we add, we will not be able to process more transactions per seconds.



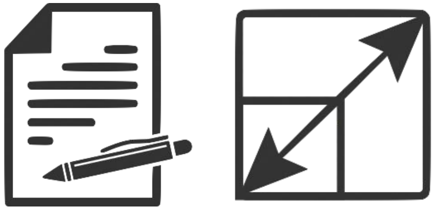


Introduction

- **What is chainspace?**

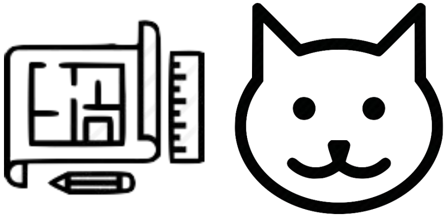
contribution I

Scalable smart contract platform



contribution II

Supporting privacy

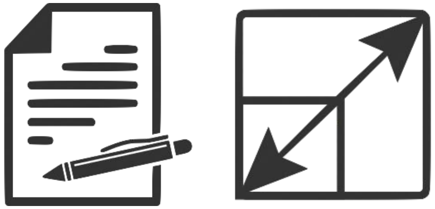


Introduction

- What is chainspace?

contribution I

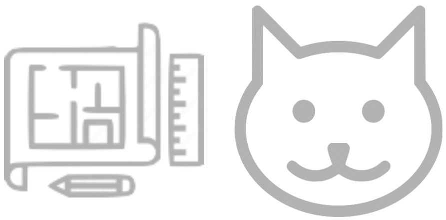
Scalable smart contract platform



contribution II

Not for today

Supporting privacy



Contents

.1.
System Overview

.2.
Scalability

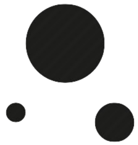
.3.
Privacy by Design

.4.
Security Properties

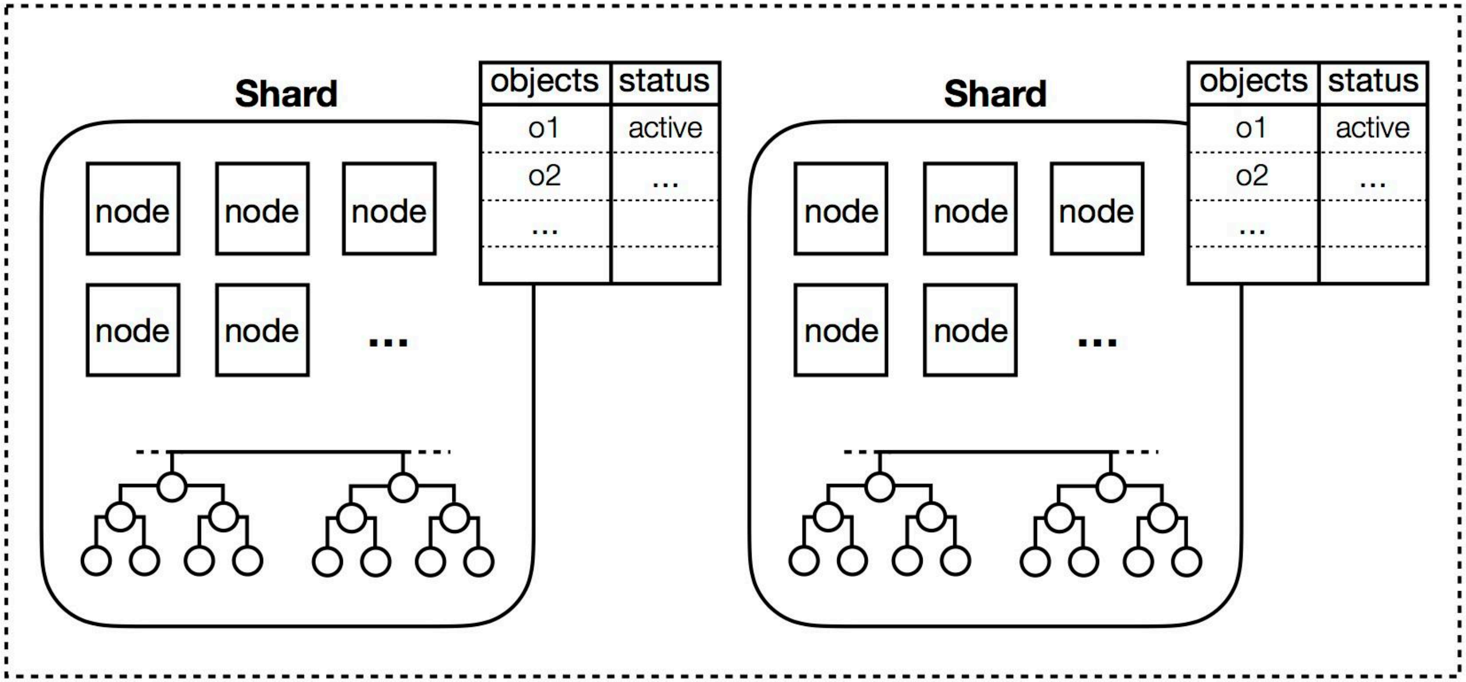
.5.
Performance

System Overview

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

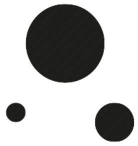


objects

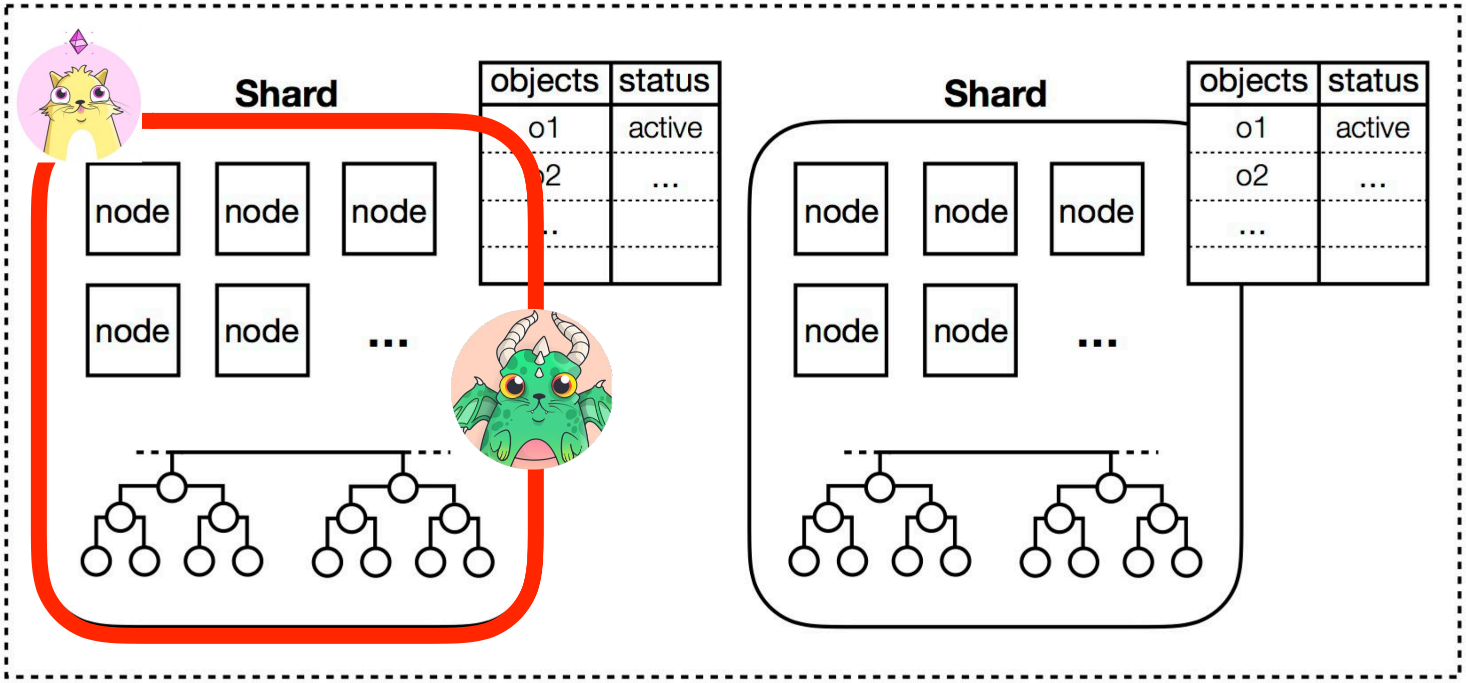


System Overview

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

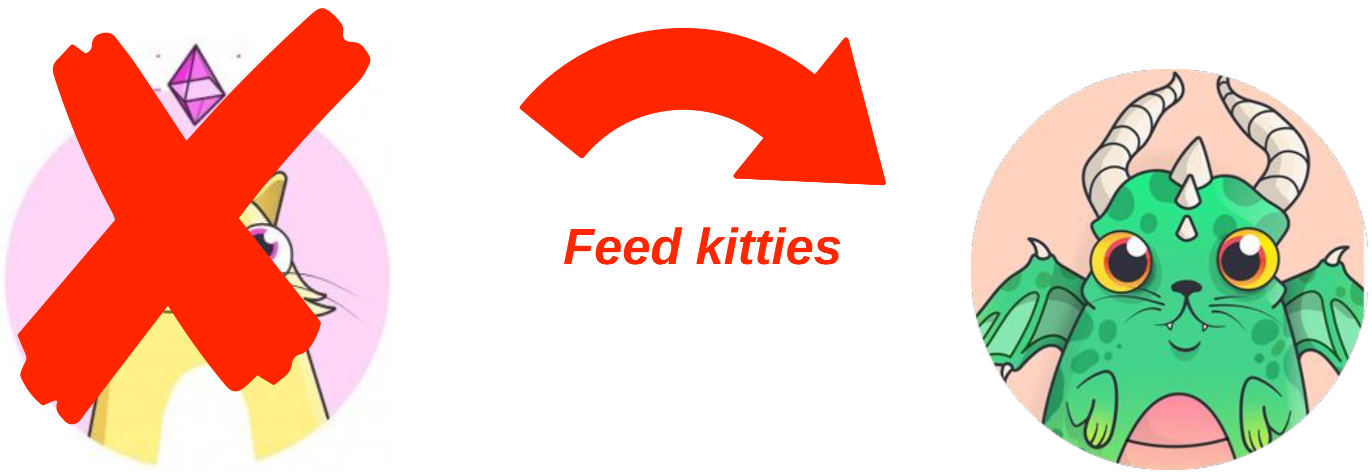


objects



System Overview

- How Chainspace works?
  - A cruel vision of it:



old object (dead)

new object (born)

Scalability

- How nodes reach consensus?

The S-BAC Protocol

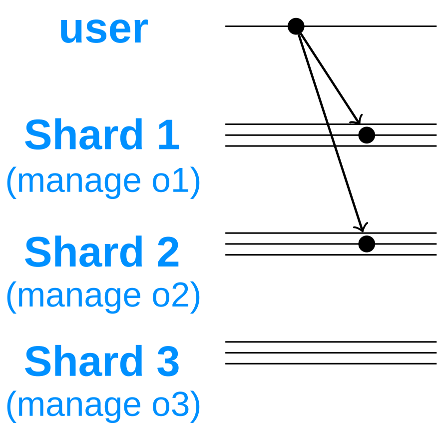
Byzantine Agreement ⊕ **Atomic Commit**

Scalability

- How nodes reach consensus?

The S-BAC Protocol

Byzantine Agreement + **Atomic Commit**

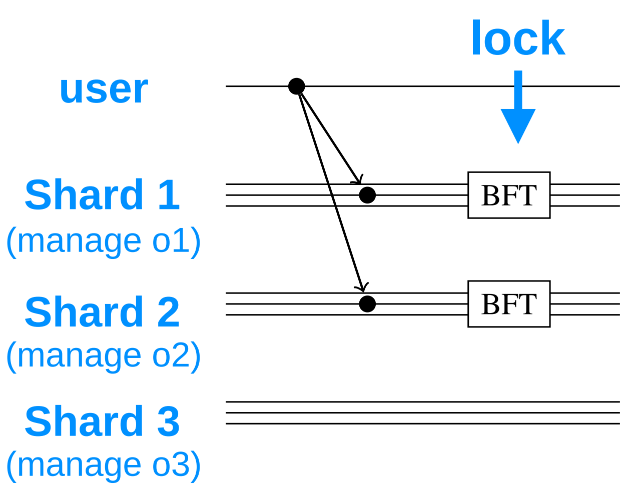


Scalability

- How nodes reach consensus?

The S-BAC Protocol

Byzantine Agreement + **Atomic Commit**

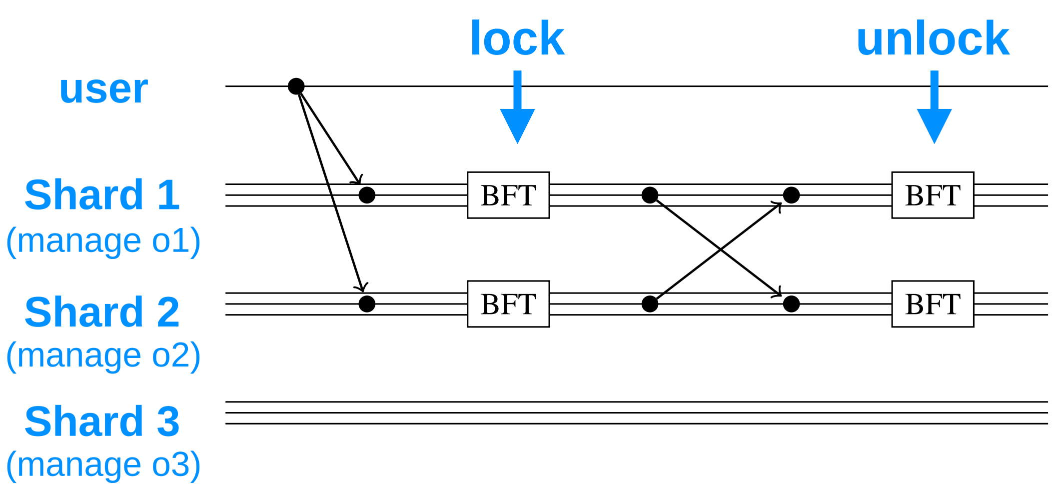


Scalability

- How nodes reach consensus?

The S-BAC Protocol

Byzantine Agreement + **Atomic Commit**

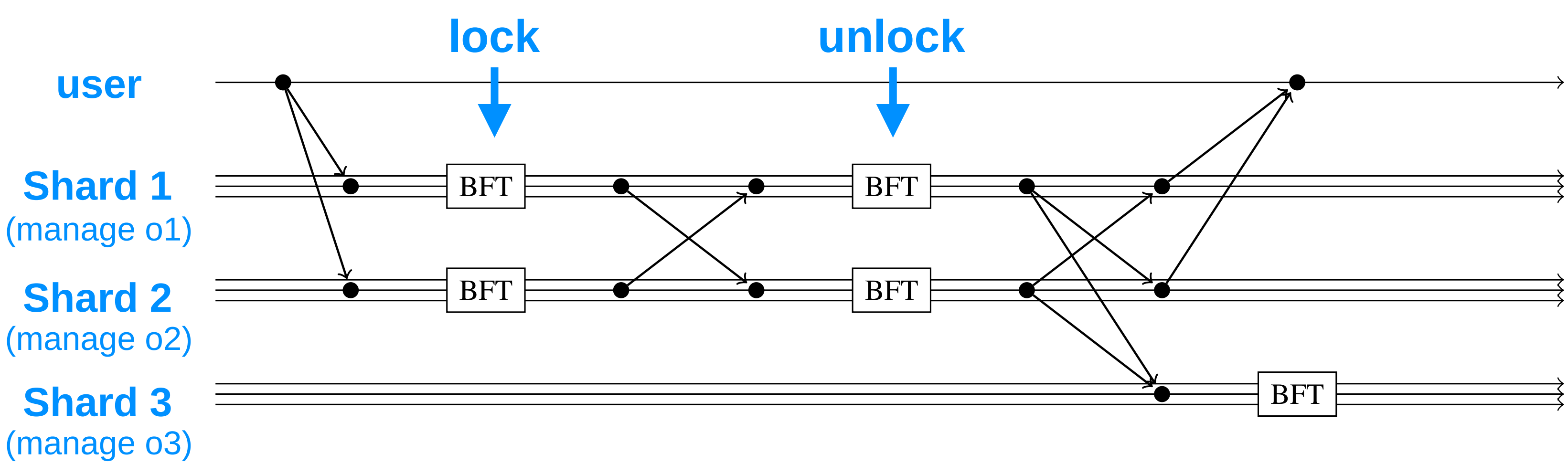


Scalability

- How nodes reach consensus?

The S-BAC Protocol

Byzantine Agreement ⊕ **Atomic Commit**

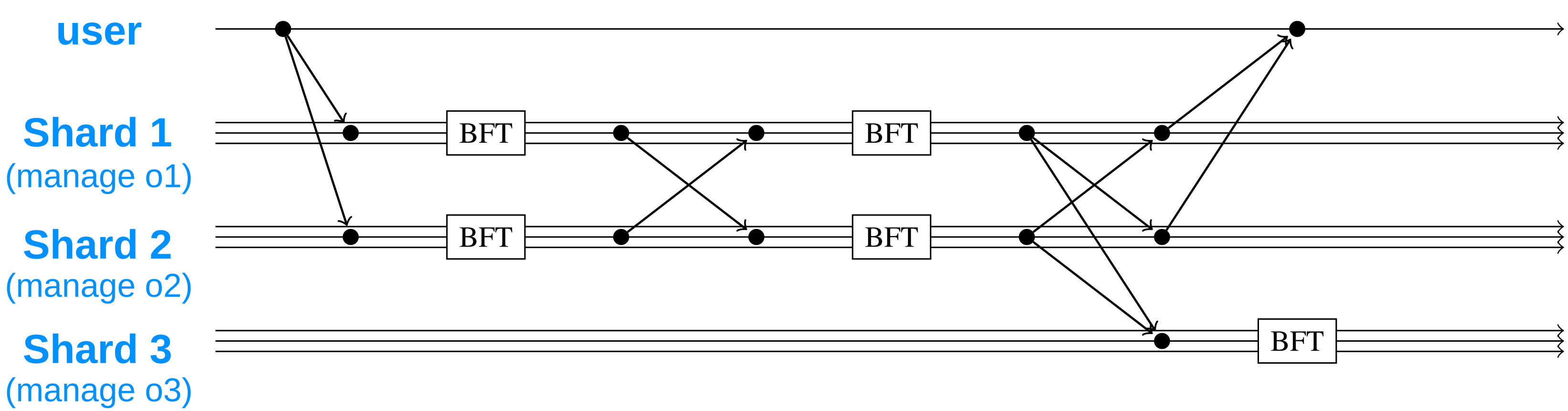


Scalability

- The wisdom behind S-BAC

Only shards managing *o1* and *o2* are reaching consensus

Shard 1 and shard 2 can work in parallel



Security Properties

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

Security Properties

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

Transparency

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

Security Properties

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

Transparency

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

Encapsulation

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

Security Properties

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

Transparency

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

Encapsulation

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

Integrity
(Honest Shard)

Only valid & non-conflicting transactions will be executed.

Security Properties

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

Transparency

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

Encapsulation

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

Integrity
(Honest Shard)

Only valid & non-conflicting transactions will be executed.

Non-Repudiation

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

Performance

- What did we implement?

Performance

- What did we implement?

Measured and tested on Amazon AWS

Performance

- What did we implement?

Measured and tested on Amazon AWS

S-BAC protocol implemented in Java

Based on BFT-SMaRt

Performance

- What did we implement?

Measured and tested on Amazon AWS

aws

S-BAC protocol implemented in Java

Based on BFT-SMaRt

Python contract simulator

Helps developers
Simulation of the checker
No need for full deployment

Performance

- What did we implement?

Measured and tested on Amazon AWS

S-BAC protocol implemented in Java

Based on BFT-SMaRt

Python contract simulator

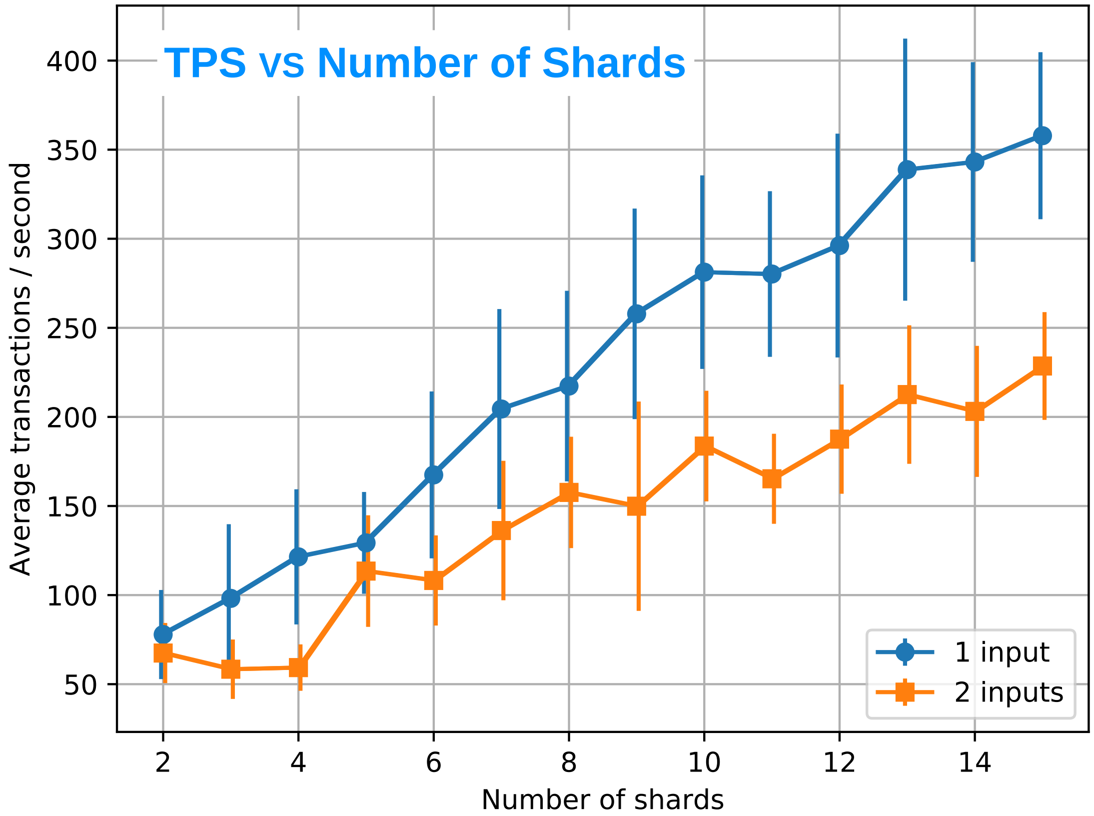
Helps developers
Simulation of the checker
No need for full deployment

Everything is released as open source software

`https://github.com/chainspace`

Performance

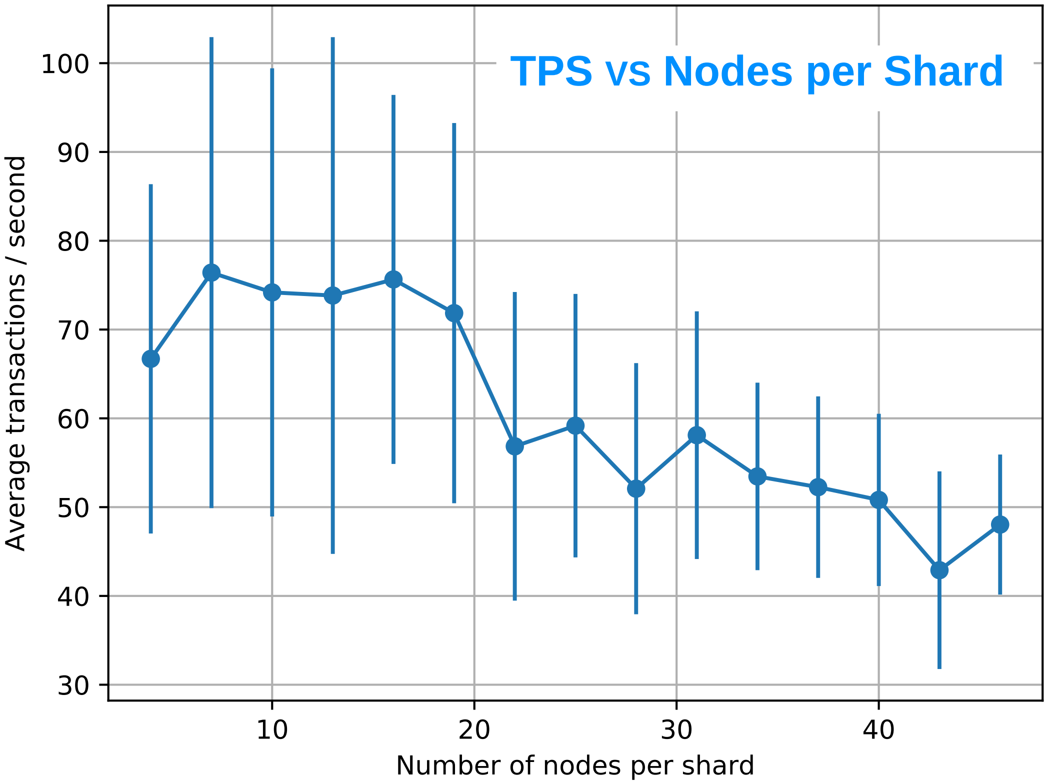
- How the number of shards influences the TPS?



TPS scales linearly with the number of shards

Performance

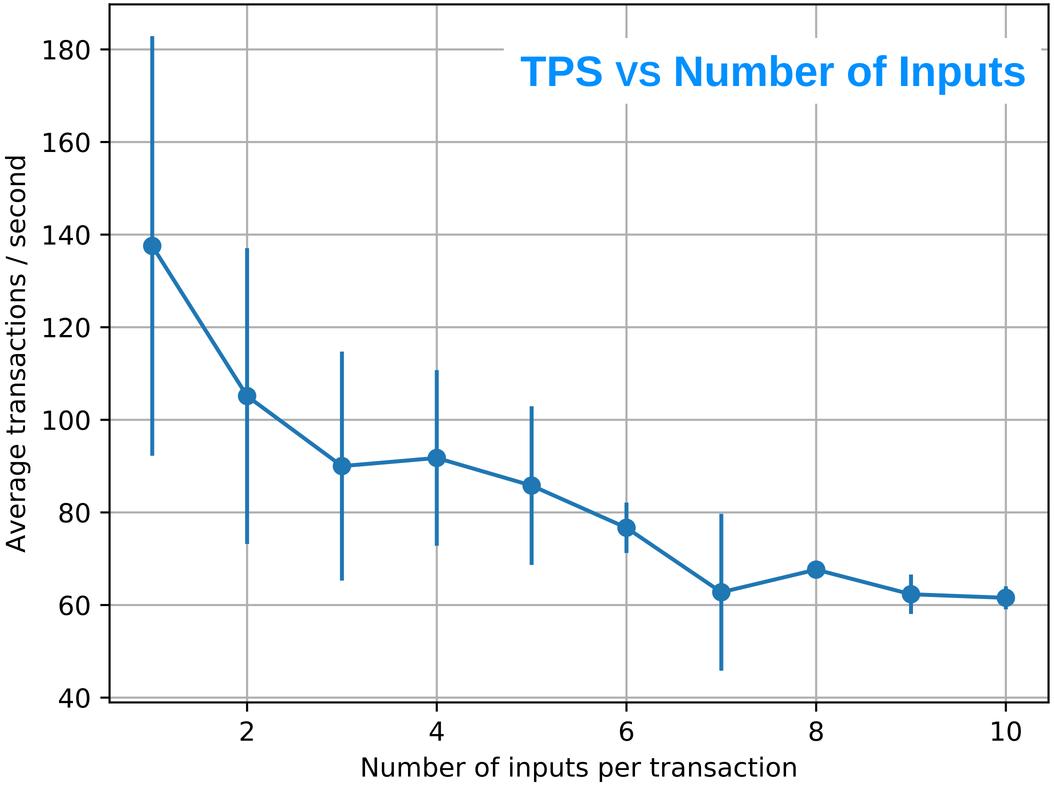
- How does the size of the shard influence the TPS?



TPS decreases slowly

Performance

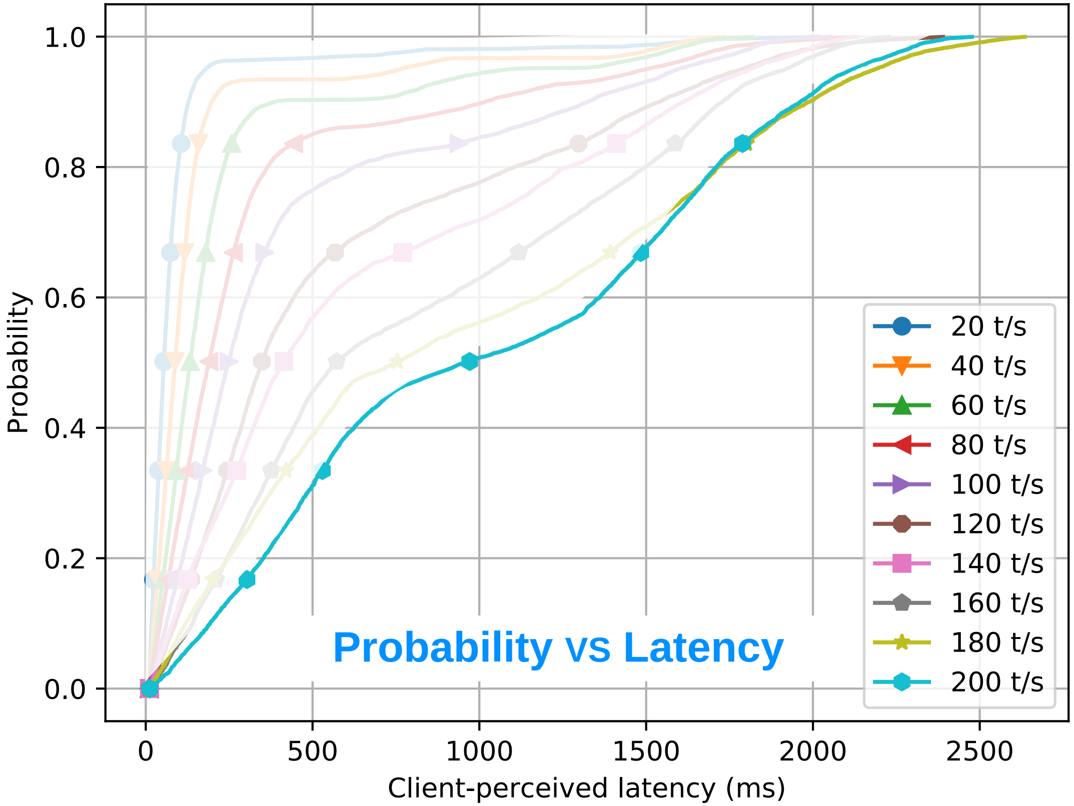
- How the number of inputs influence the TPS?



TPS decreases slowly and then flattens out

Performance

- How is the trade off between TPS and latency?



Low latency even when the system is heavy loaded

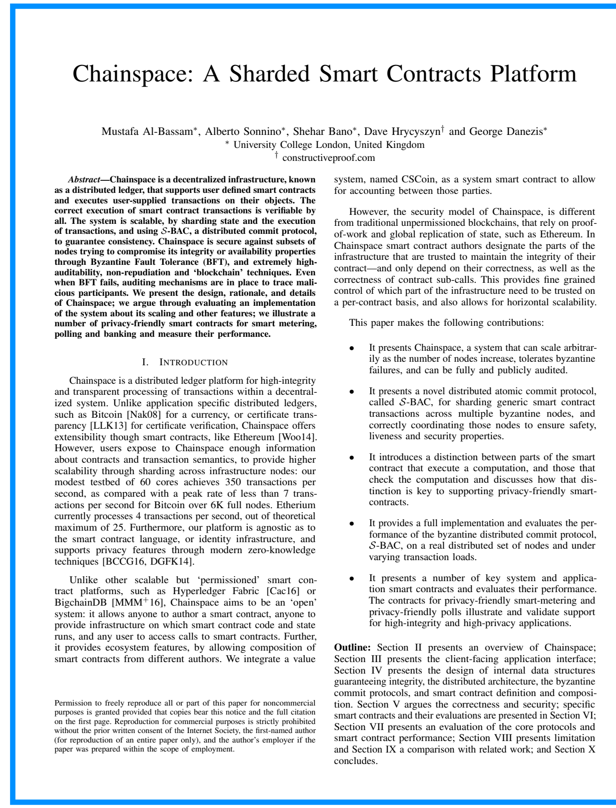
What else is in the paper?

Cross shard transactions

Smart metering contract

Platform for decision making

contracts benchmarking and evaluation



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 $\mathcal{S}$-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.

Permission to freely reproduce all or part of this paper for noncommercial purposes is granted provided that copies bear this notice and the full citation on the first page. Reproduction for commercial purposes is strictly prohibited without the prior written consent of the Internet Society, the first-named author (for reproduction of an entire paper only), and the author's employer if the paper was prepared within the scope of employment.

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 $\mathcal{S}$-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, $\mathcal{S}$-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.

Future Works

1. **How to recover from malicious shards?**

Future Works

1. How to recover from malicious shards?

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

Future Works

1. How to recover from malicious shards?

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

3. How to configure shards?

Future Works

1. How to recover from malicious shards?

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

3. How to configure shards?

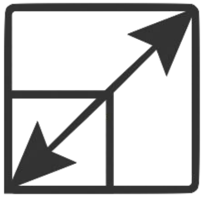
4. How to incentivise nodes?

Conclusions

- What did we talked about ?

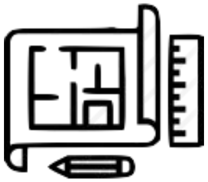
contribution I

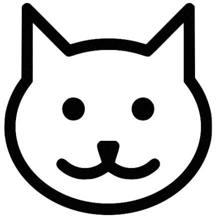
Scalable smart contract platform



contribution II

Supporting privacy





Conclusions

- **Main take-aways**

sharding → **scalability**

In the paper

execution / checker → **privacy by design**

UCL

Thank you for your attention

Questions?

Alberto Sonnino

alberto.sonnino@ucl.ac.uk

https://sonnino.com

https://github.com/chainspace

EPSRC Engineering and Physical Sciences Research Council

decode

THE ALAN TURING INSTITUTE

This work is supported in part by EPSRC Grant EP/M013286/1, the EU H2020 DECODE project (grant agreement number 732546), and The Alan Turing Institute.