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.
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.
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…
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

Supporting privacy

Not for today
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
System Overview

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

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

old object (dead)  ▶️  Feed kitties  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

Shard 1 (manage o1)
Shard 2 (manage o2)
Shard 3 (manage o3)

user

From the point of view of scalability, Chainspace capacity can be managed by different nodes within the system to ensure that load of accepting transactions is distributed across them. Furthermore, those inputs must have on average a constant, or sub-linear, number of inputs and references (see Figure 6). Furthermore, those inputs must have on average a constant, or sub-linear, number of inputs and references (see Figure 6).
Scalability

How nodes reach consensus?

The S-BAC Protocol

Byzantine Agreement + Atomic Commit

user

lock

Shard 1
(manage o1)

Shard 2
(manage o2)

Shard 3
(manage o3)
Scalability

- How nodes reach consensus?

The S-BAC Protocol

Byzantine Agreement + Atomic Commit

user

Shard 1
.manage o1

Shard 2
.manage o2

Shard 3
.manage o3

lock

unlock
Scalability

How nodes reach consensus?

The S-BAC Protocol

Byzantine Agreement + Atomic Commit

Scalability

From the point of view of scalability, Chainspace capacity involves a fixed number of operations. Each transaction proposed by a user, and no locks are held indefinitely on input objects, only one of them is accepted by all nodes. This is achieved through the BFT protocol that is idempotent. Specifically, the BFT-Initiator drives the composed S-BAC protocol to account for network losses. Next they proceed to wait; if no reminders to the BFT-Initiator along with the original message are received, they time out again, other nodes perform the action of BFT-Initiator to act on it until they time out. They first send a BFT-Reminder to the BFT-Initiator along with the original message. The BFT-Initiator then sends requests to the concerned shards to create the objects. On receipt, each shard assesses whether overall 'All proposed(accept, T)' or 'Some proposed(abort, T)' holds across shards, sequences the accept(T,*) and sends the decision to the user. All cross-shard arrows represent a multicast of all nodes in one shard to all nodes in another.
Scalability

• The wisdom behind S-BAC

Only shards managing o1 and o2 are reaching consensus

Shard 1 and shard 2 can work in parallel

- user
- Shard 1 (manage o1)
- Shard 2 (manage o2)
- Shard 3 (manage o3)
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.

  - **Integrity**
    (Honest Shard)
    Only valid & non-conflicting transactions will be executed.

  - **Encapsulation**

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

  - **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
- 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?

![Graph showing TPS vs Nodes per Shard](image)

**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

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

Abstract—Chainspace is a distributed ledger platform for high-integrity and high-privacy applications. It provides a novel distributed atomic commit protocol, called S-BAC, for sharding complex smart contracts, which are composed of 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 observe the computation and discuss 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 properties of the system about its scaling and other features; we illustrate a prototype platform with detailed measurements of real-world performance, testing, polling and banking and measure their performance.

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 above. Reproduction for commercial purposes is strictly prohibited on the first page. Reproduction for commercial purposes is granted provided that copies bear this notice and the full citation above. The paper was prepared within the scope of employment. It presents a novel distributed atomic commit protocol, called S-BAC, for sharding complex smart contracts, which are composed of 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 observe the computation and discuss 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 properties of the system about its scaling and other features; we illustrate a prototype platform with detailed measurements of real-world performance, testing, polling and banking and measure their performance.

I. INTRODUCTION

Chainspace is a distributed ledger platform for high-integrity and high-privacy applications. It provides a novel distributed atomic commit protocol, called S-BAC, for sharding complex smart contracts, which are composed of 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 observe the computation and discuss 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 properties of the system about its scaling and other features; we illustrate a prototype platform with detailed measurements of real-world performance, testing, polling and banking and measure their performance.

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 above. Reproduction for commercial purposes is strictly prohibited on the first page. Reproduction for commercial purposes is granted provided that copies bear this notice and the full citation above. The paper was prepared within the scope of employment. It presents a novel distributed atomic commit protocol, called S-BAC, for sharding complex smart contracts, which are composed of 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 observe the computation and discuss 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 properties of the system about its scaling and other features; we illustrate a prototype platform with detailed measurements of real-world performance, testing, polling and banking and measure their performance.

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 for accounting between those parties. Section V argues the correctness and security; specifically, proving that the system is correct and executes user-supplied transactions on their objects. 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 properties of the system about its scaling and other features; we illustrate a prototype platform with detailed measurements of real-world performance, testing, polling and banking and measure their performance.

Section VI is a comparison with related work, and Section VII 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

- execution / checker
- privacy by design

In the paper
Thank you for your attention
Questions?

Alberto Sonnino
alberto.sonnino@ucl.ac.uk
https://sonnino.com

https://github.com/chainspace

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.