

Chainspace: A Sharded Smart Contract Platform

Authors

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



* University College London ** constructiveproof.com



■ Blockchains are cool — but scale badly



■ Blockchains are cool — but scale badly





Blockchains are cool — but scale badly







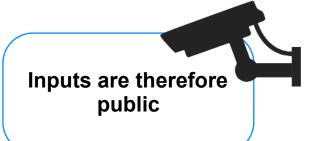
Blockchains are cool — but scale badly





Hard to operate on secret inputs







Related works

	Smart Contract	Scalable	Privacy
Ethereum		X	X
Hawk		X	
ZCash		X	
Omniledger			X
RSCoin			X



Introduction

■ What is chainspace?

contribution I

Scalable smart contract platform







Introduction

■ What is chainspace?

contribution I

Scalable smart contract platform





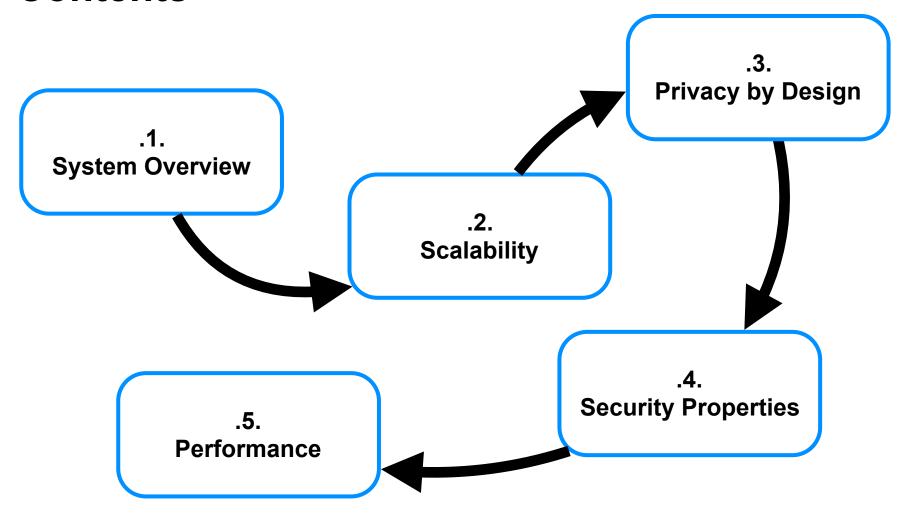
contribution II

Supporting privacy





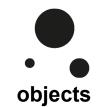
Contents

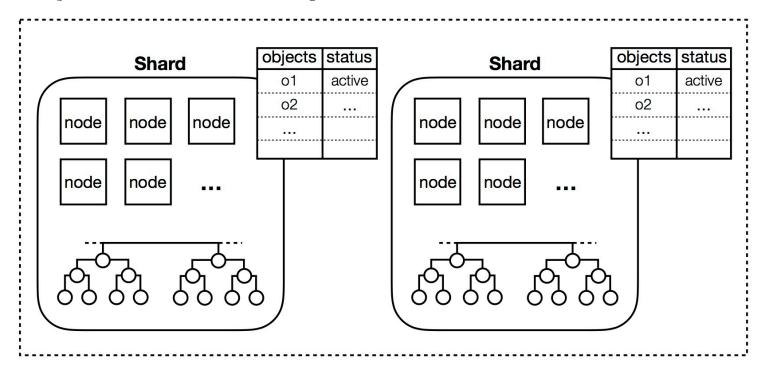




System Overview

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







How nodes reach consensus?

The S-BAC Protocol

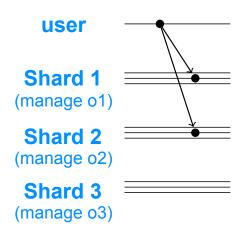
Byzantine Agreement



Atomic Commit

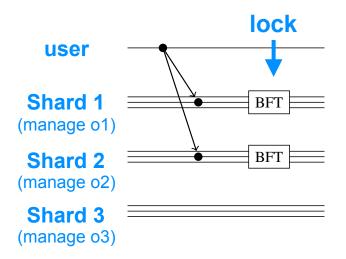






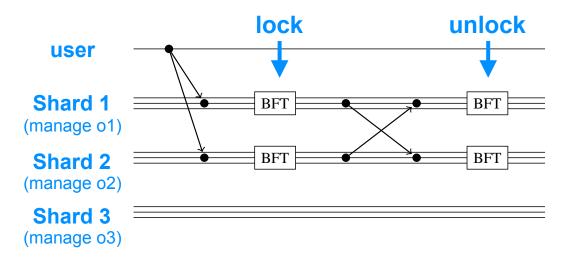




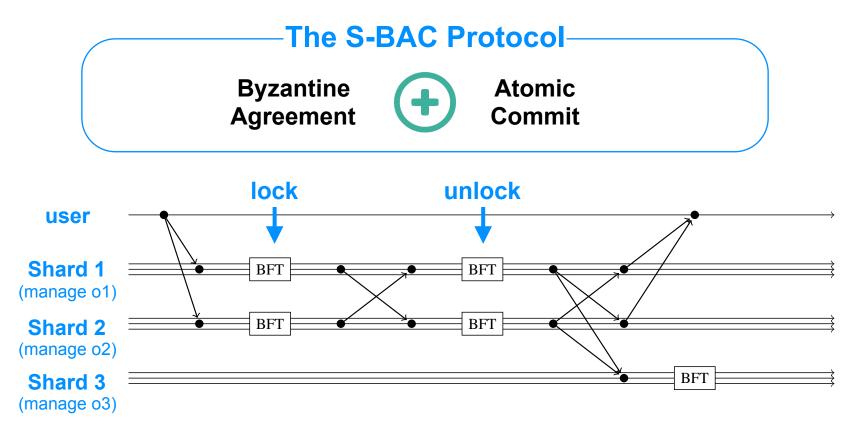










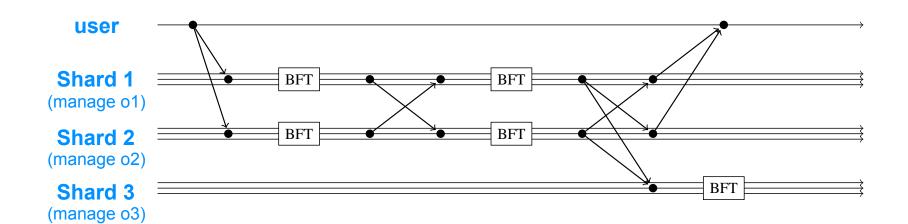




The Wisdom behind S-BAC

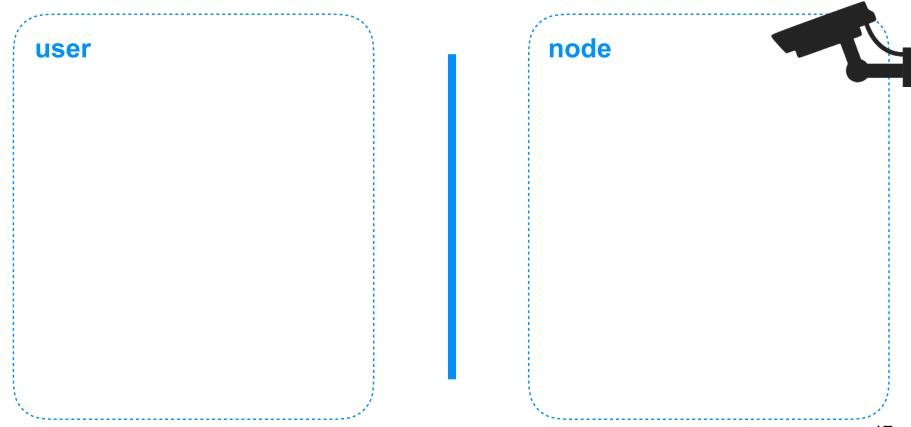
Only shards managing *o1* and *o2* are reaching consensus

Shard 1 and shard 2 can work in parallel





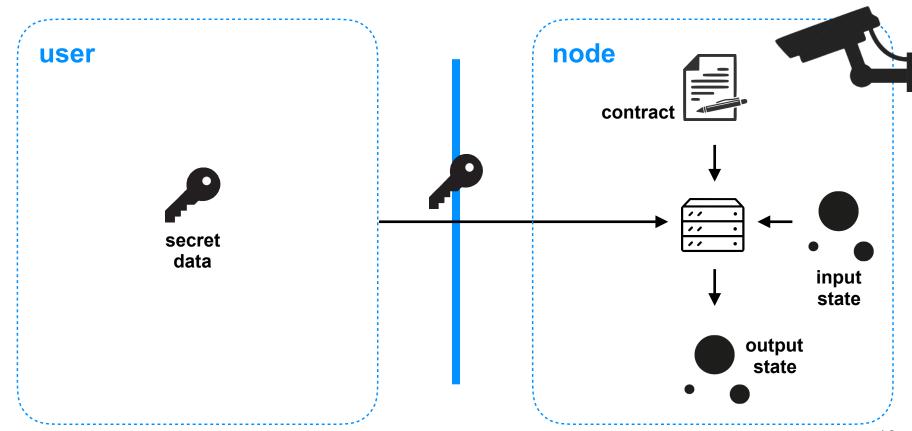
Transaction in classic blockchains



17



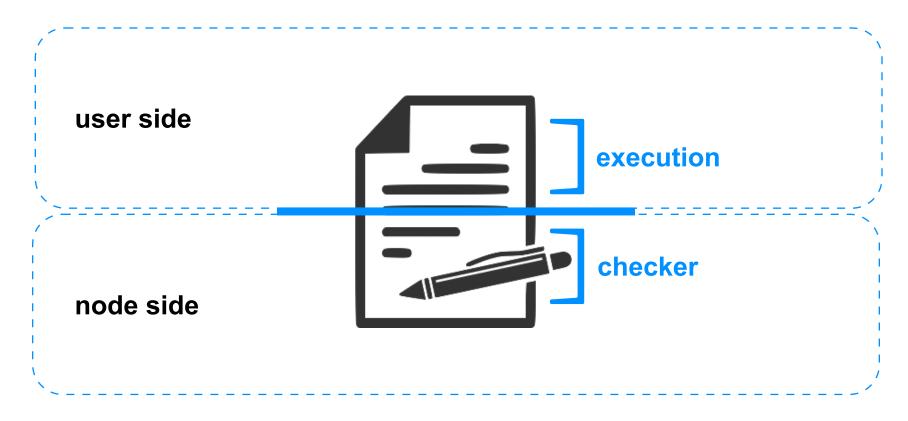
Transaction in classic blockchains



18

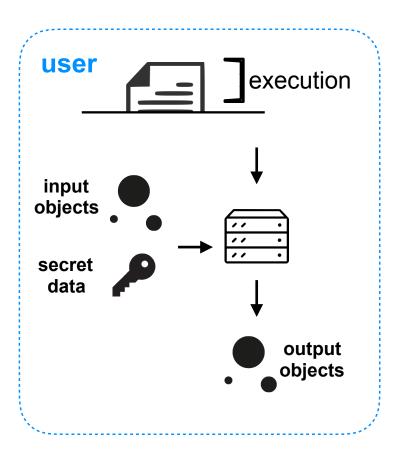


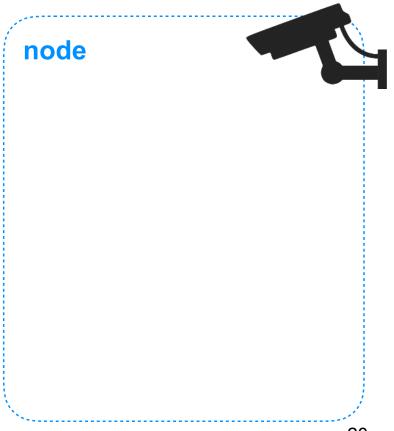
What are Chainspace Smart Contracts?





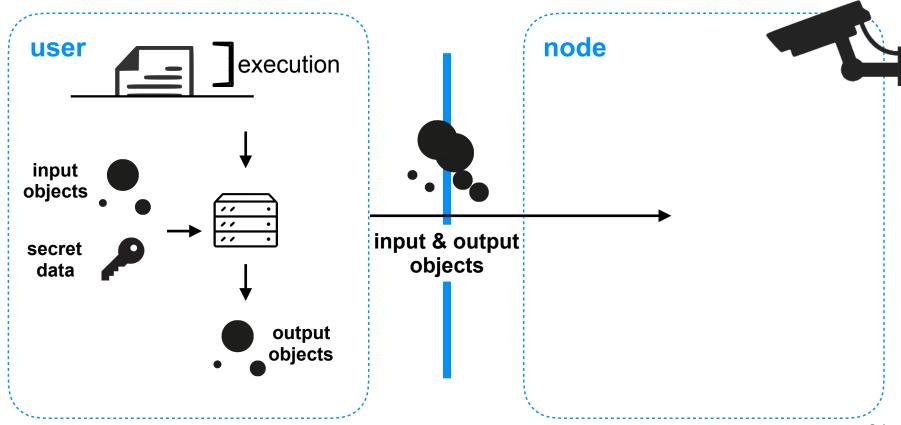
Chainspace transaction





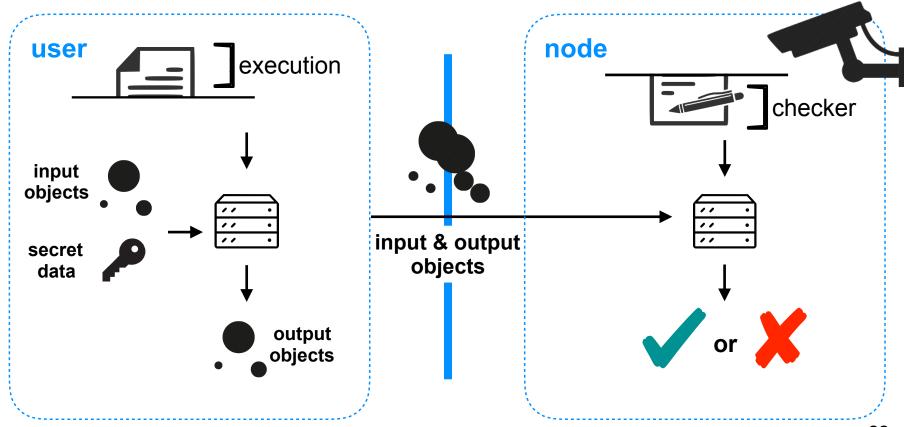


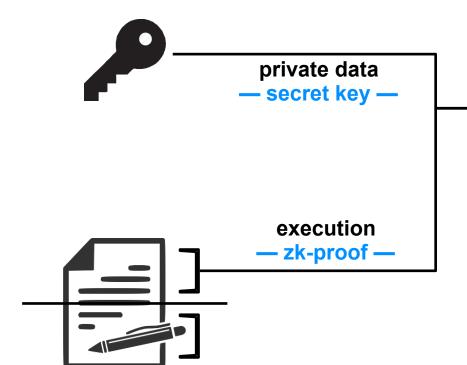
Chainspace transaction





Chainspace transaction





$$\pi = \mathrm{PK}\{(k) : \nu = \left(g_1^k\right)^{UUID} \ \land \ \kappa = lpha eta^k\}$$

Adding ν to L prevent a citizen to vote twice during the sa campaign (prevent double spending), while the proof π ensu that ν has been built from a signed private key k.

C. Mapping authorities to blockchain nodes

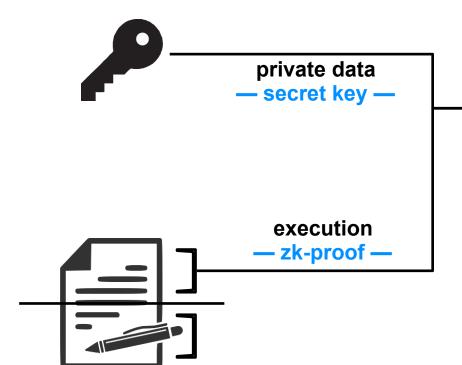
Alberto: @George, Describe how the CoCoNut authori can also be Chainspace nodes (to make clear the poten Private data never leave the chelogram when deeply related to blockchains), since actually built all of this to have credentials in smart contra

> V. EVALUATIONSET A. Primitives evaluation

The signature scheme as been impremented in pyth using the two crypo libraries petlib [1] and bplib [2]. bilinear pairing works over a Barreto-Naehrig [17] curve, us OpenSSL as the arithmetic backend. We have released the co as an open-source project on GitHub.³.

Table I shows the mean (μ) and standard deviation $(\sqrt{\sigma^2})$ the execution of each procedure described in section section Each entry is the result of 10,000 measured on an Octa-c Dell desktop computer, 3.6GHz Intel Xeon. This table sho that signing is much faster than verifying signatures (ab 15 times faster for the scheme working on clear message

can also Private data never leave the chelonicocon



compute a value ν as follows. $\pi =$

UUID

A. Primitives evalued $i \overline{o} \overline{n}$

Adding ν campaign

that ν has

C. Mappi Alberto.

actually b

A. Primite

Then. They signature is solution that obtaining the listusingarthebtsild aryppolkhravitsdgetliboolf] s buildliftean phiringnworkshower ao Bahreitoe Nealth

$$\pi = \mathrm{PK}\{(k) : \nu = \left(g_1^k\right)^{UUID} \ \land$$

Adding ν to L prevent a citizen to vote twice campaign (prevent double spending), while the that ν has been built from a signed private

C. Mapping authorities to blockchain nodes

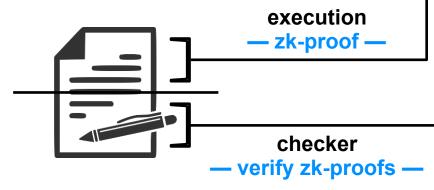
Alberto: @George, Des sibe how the Co can also be Chainspace Les (to make c The si⊊ using the two crypo libraties, petlib 1, and bplib [2]. bilinear pairing works over Barreto-Naehrig [17] curve, us OpenSSL as the arithmetic backend. We have released the co as an open-source project on GitHub/3. EVALUATION

Table I shown the mann ω condition ($\sqrt{\sigma^2}$ the execution of 2ch procedure described in section section he signature scheme has been implement the entry is the result of 10,000 measured on an Octacusing the two crypo libraries petlib [1] at Dell desktop computer, 3.6GHz Intel Xeon This table shows that standard pairing works over a Barreto-Naehrist that standard pairing w that signing is mu 15 times farer for the s

> Table I shows the mean would standard do the execution of each procedure described in Each entry it the result of 10,000 measured Dell desktop computer, 3.6GHz Intel Xeon. that signing is much faster than verifying 15 times faster for the scheme working or

Private data never leave the





tun aiso-be enumspate nowes](whake ciear me poiemui the orcores oparatives holds the citizen's byotesh limitial tredet zerodululythericase of thas petition the leantions are and only not least and NO: and the felds owner and verifier respectively hold the public key of the third party creating the petition and of 1) authorities issuing the credentials. In order to sign the petition, the users compute a value ν as follows.

EVALUATION

A. Primitives evaludti \overline{on} $(g_1^k)^{UUID}$

Then, They signature is solven to the obtain symptoment of toin symptoment of the obtain symptoment of the obtains and the obtaining the obtai listuLingantdebtiild aryppoolkhravitsdgetliboobf] shootvibglibhall v The buildliftean phiringmental wer an Bahreitoe Nathmin [st. 7] curve, using

$$\pi = \mathrm{PK}\{(k) :
u = \left(g_1^k\right)^{UUID} \ \wedge \ \kappa = lphaeta^k\}$$

Adding ν to L prevent a citizen to vote twice during the same campaign (prevent double spending), while the profession π ensures that ν has been built from a signed private key

C. Mapping authorities to blockchain nodes

Alberto: @George, Describe how the CoCoNut authorities can also be Chainspace nodes (to make clear the potential actually

that signing is mu A. Primutveinesulfactor the s

The signature so nemerator below wireplane with the dard de using the two crypt libraries until offel can stock and less the din bilinear pairing works Frech Barrette-Nachriga (t. 17) cur, 000 using sure OpenSSL as the arithmental value released the collection. as an open-source projection GitHubs much faster than verifying

Table I shows the mean (time and assended a derei as to be in a first one of the control of the the execution of each procedure described in section section 25. Each entry is the result of 10,000 measured on an Octa-core Dall dealston computer 2 6CHz Intel Voon This table shows



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



- 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).



- 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).



- 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.



What did we implement?





What did we implement?







What did we implement?







S-BAC protocol implemented in Java

Based on BFT-SMaRt



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



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

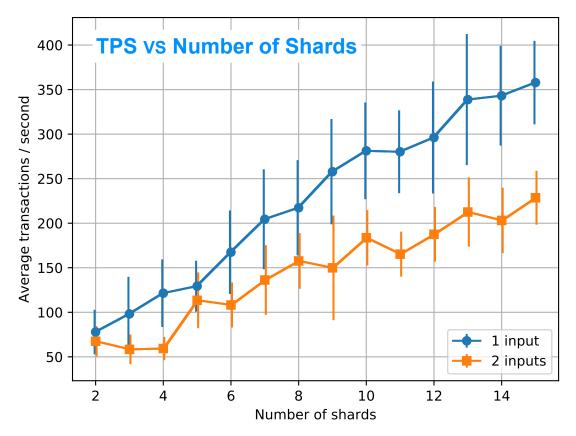
Everything is released as open source software

https://github.com/chainspace





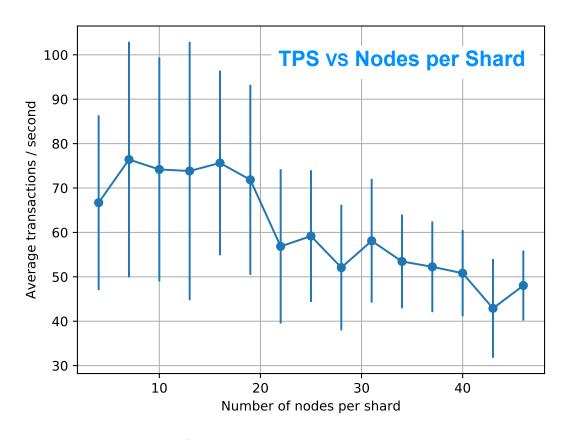
How the number of shards influences the TPS?



TPS scales linearly with the number of shards



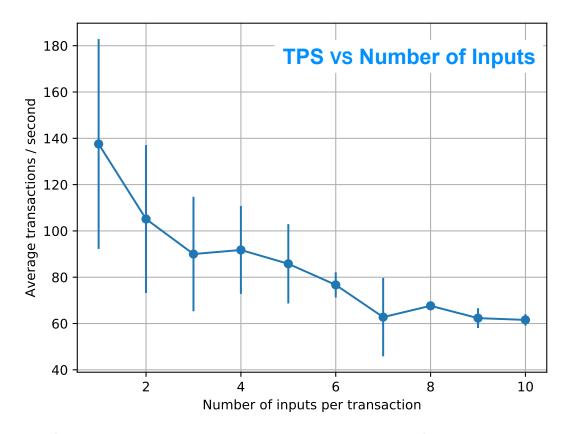
How does the size of the shard influence the TPS?



TPS decreases slowly



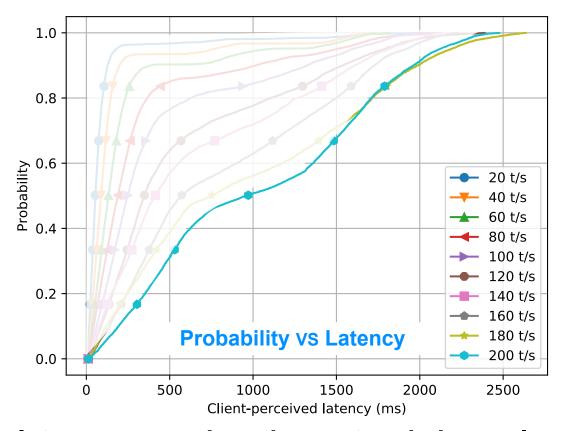
How the number of inputs influence the TPS?



TPS decreases slowly and then flattens out



How is the trade off between TPS and latency?



Low latency even when the system is heavy loaded

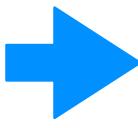


What else is in the paper?

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

† constructivencyof.com

Advance—Chaisquae is a decentralized infrastructure, known as distributed leiger, that supports user offende mant contracts and executes user-supplied transactions on their objects. The correct execution of mart contract transactions is verifiable by all. The system is scalable, by duarding state and the execution to guarantee considers, Chaisquae is secure against subsets of nodes trying to compromise its integrity or availability properties frough Byzandies Fault Toderace (BFT), and externelly high-auditiability, mon-epudation and 'blockchair techniques. Even consideration of the system about the state of Chaisquae, Even Comparison, the properties of the system about the scale and other contents of the system about its scaling and other features; well historiate a number of privacy-friendly mart contracts for mant metering, public and backless and unconstructive proferomance.

I. Introductio

Chainspace is a distributed ledger platform for high-inegrity and transparent processing of transactions within a decentralized system. Unlike application specific distributed ledgers, such as Bitcoin [Ma80] for a currency, or certificate transparency [LLK13] for certificate verification. Chainspace offers the contract of the contra

Unlike other scalable but 'permissioned' smart contract platforms, such as Hyperdedger Fabric [Cac61] or BigchainDB [MMM*16], Chainsquee 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

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 to the first page. Reproduction for commercial purposes is strictly probibiled without the prior written consent of the Internet Society, the first anneal author (for reproduction of an entire paper only), and the author's employer if the paper was required within the scope of employment.

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 block-tains, that rely on proofof-work and global replication of state, such as Ethereum. In Chainspace sums contract authors designate the parts of the infrastructure that are trusted to maintain the integrity of their infrastructure that are trusted to maintain the integrity of their contractions of the 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 smartcontracts.
- 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 Chairspace; Section III presents the client-fining application interface; Section IV presents the design of internal data structures guaranteein 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; smart potentary 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?

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

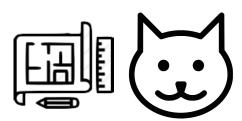
Scalable smart contract platform





contribution II

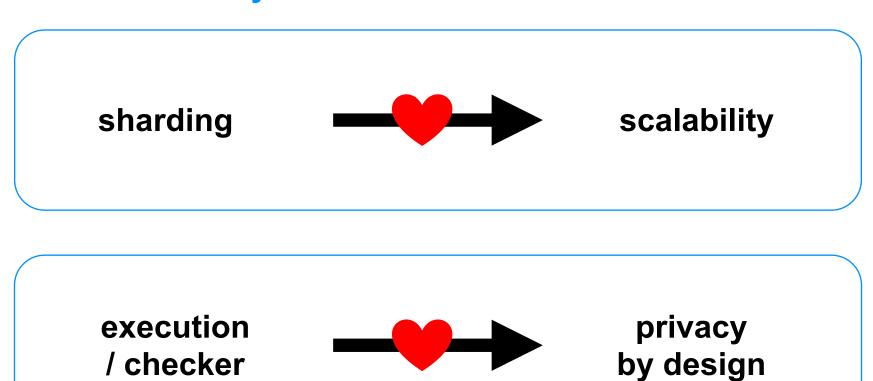
Supporting privacy





Conclusions

Main take-aways





Thank you for your attention Questions?

Alberto Sonnino alberto.sonnino@ucl.ac.uk

https://sonnino.com



https://github.com/chainspace





