

# **Coconut:** Threshold Issuance Selective Disclosure Credentials with Applications to Distributed Ledgers

## **Authors**

**Alberto Sonnino\***

Mustafa Al-Bassam\*

Shehar Bano\*

Sarah Meiklejohn\*

George Danezis\*

\* University College London

---

February 2019

# The Authors



**Alberto Sonnino**



**Mustafa Al-Bassam**



**Bano Shehar**



**Sarah Meiklejohn**



**George Danezis**

# Challenges in blockchains



**Strong integrity**

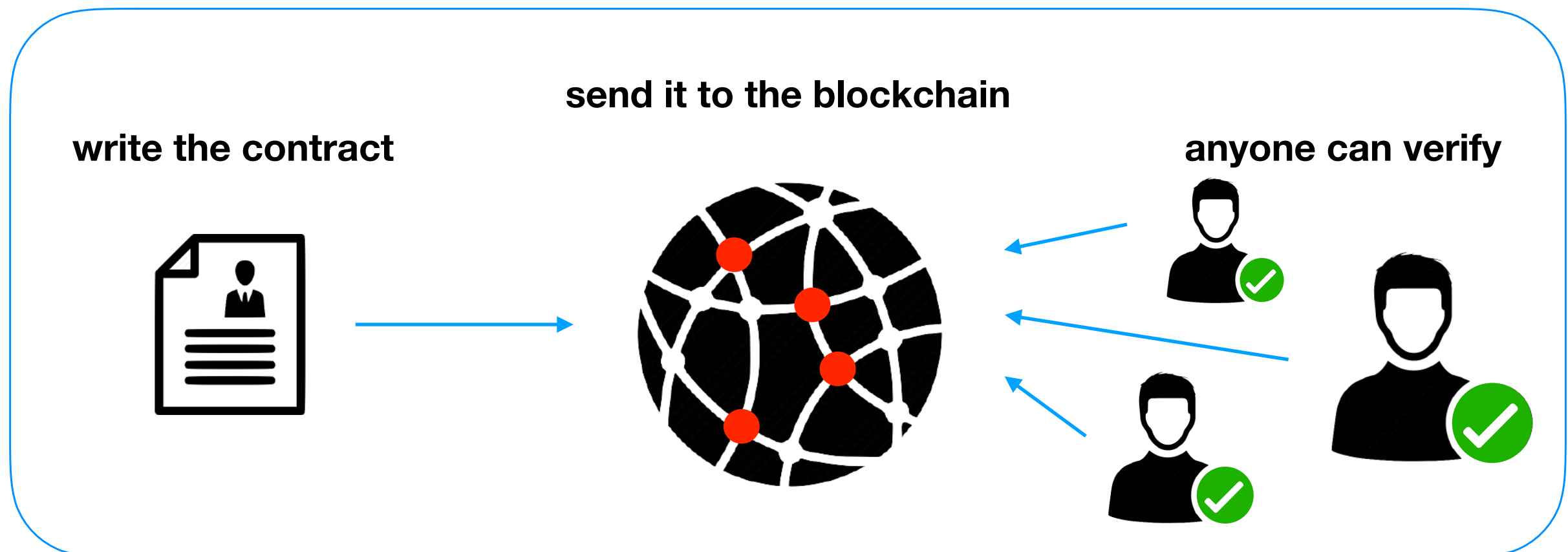


**Poor privacy**

# Challenges in blockchains

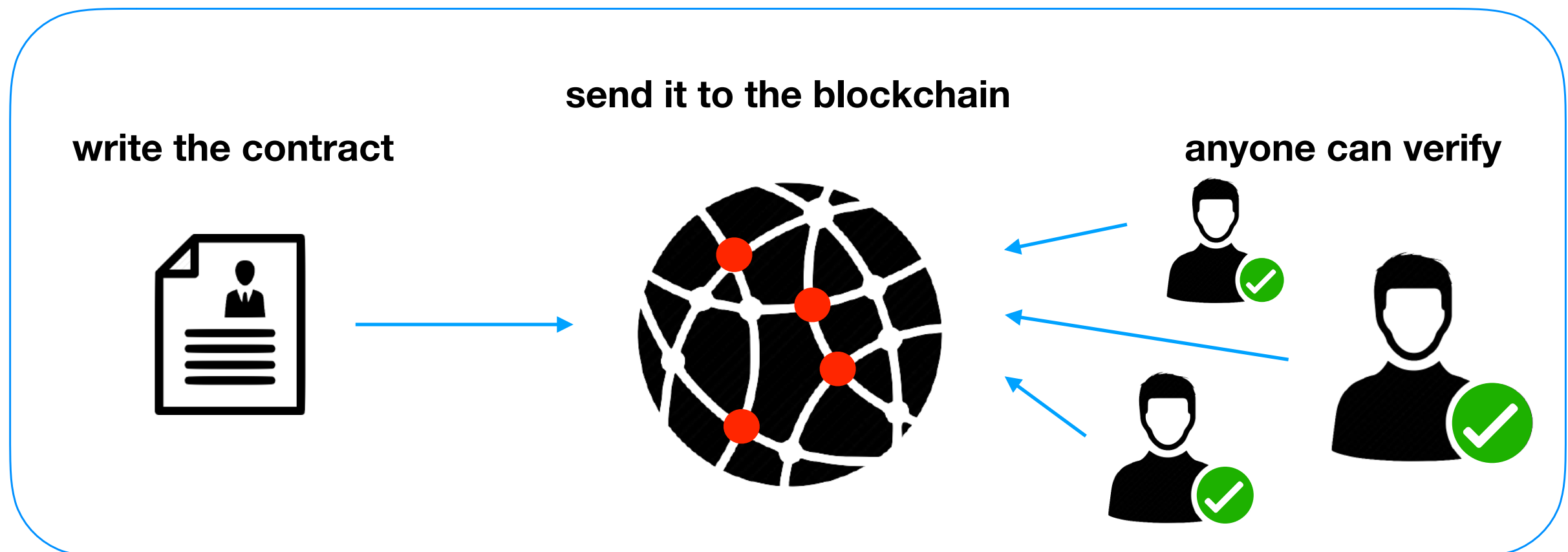
✓ **Strong integrity**

✗ **Poor privacy**



# Challenges in blockchains

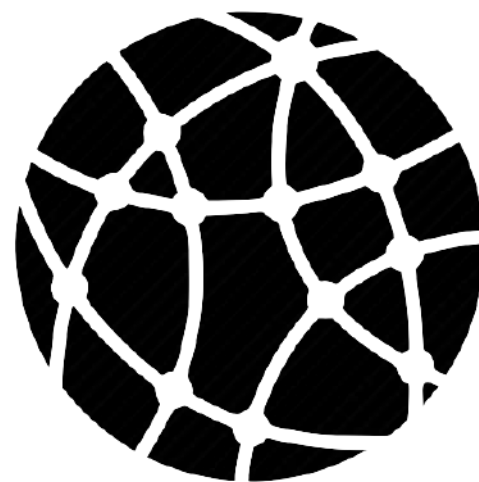
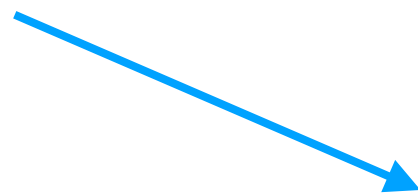
Can we issue credentials in this setting?



# What are we trying to do?

- Issuing credentials through smart contracts

write the contract



... while preserving privacy

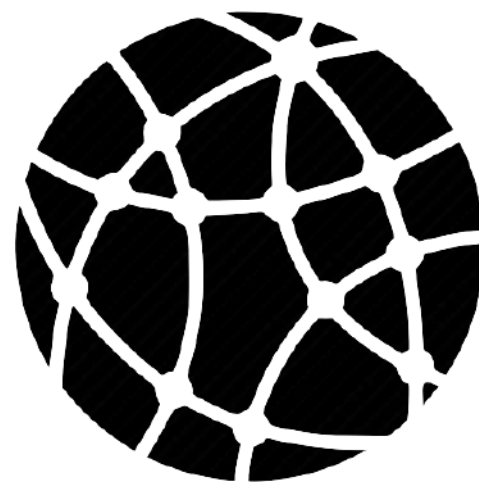
# What are we trying to do?

- Issuing credentials through smart contracts

write the contract



some attributes



... while preserving privacy

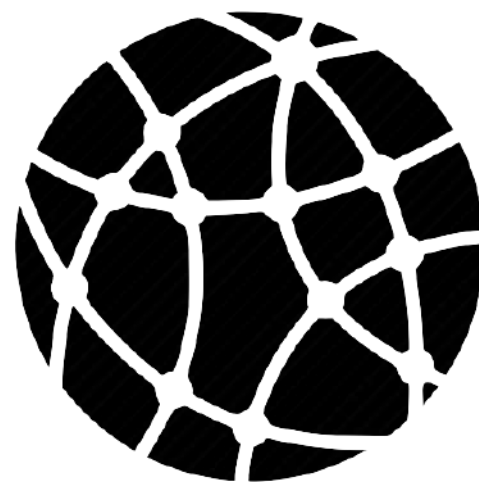
# What are we trying to do?

- Issuing credentials through smart contracts

write the contract



some attributes



credentials

... while preserving privacy



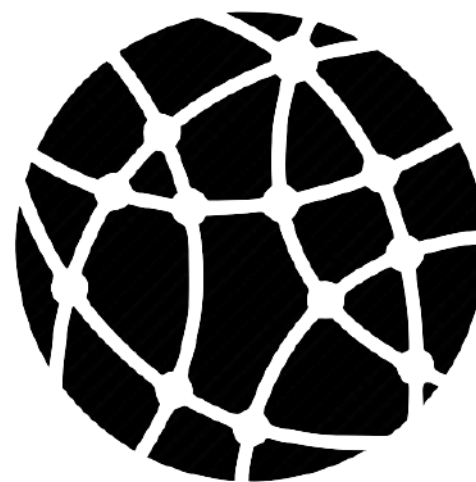
# What are we trying to do?

- Issuing credentials through smart contracts

another contract



credentials



... while preserving privacy

# What are we trying to do?

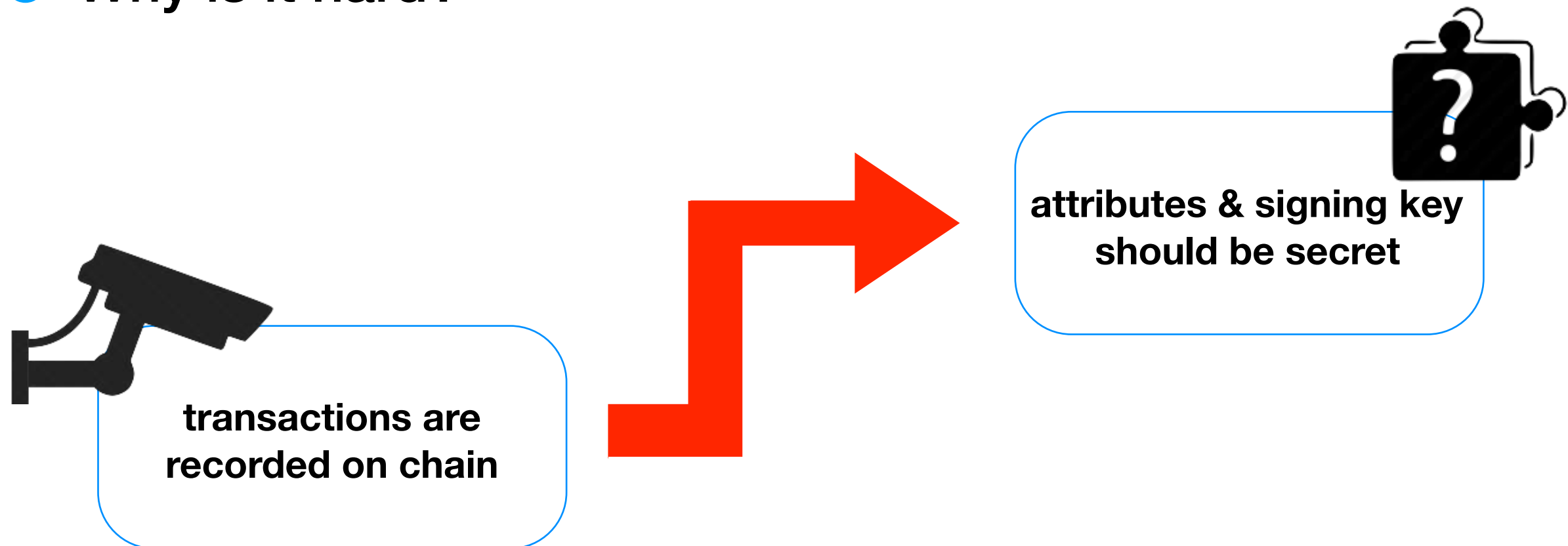
- Why is it hard?



**In a decentralised setting**

# What are we trying to do?

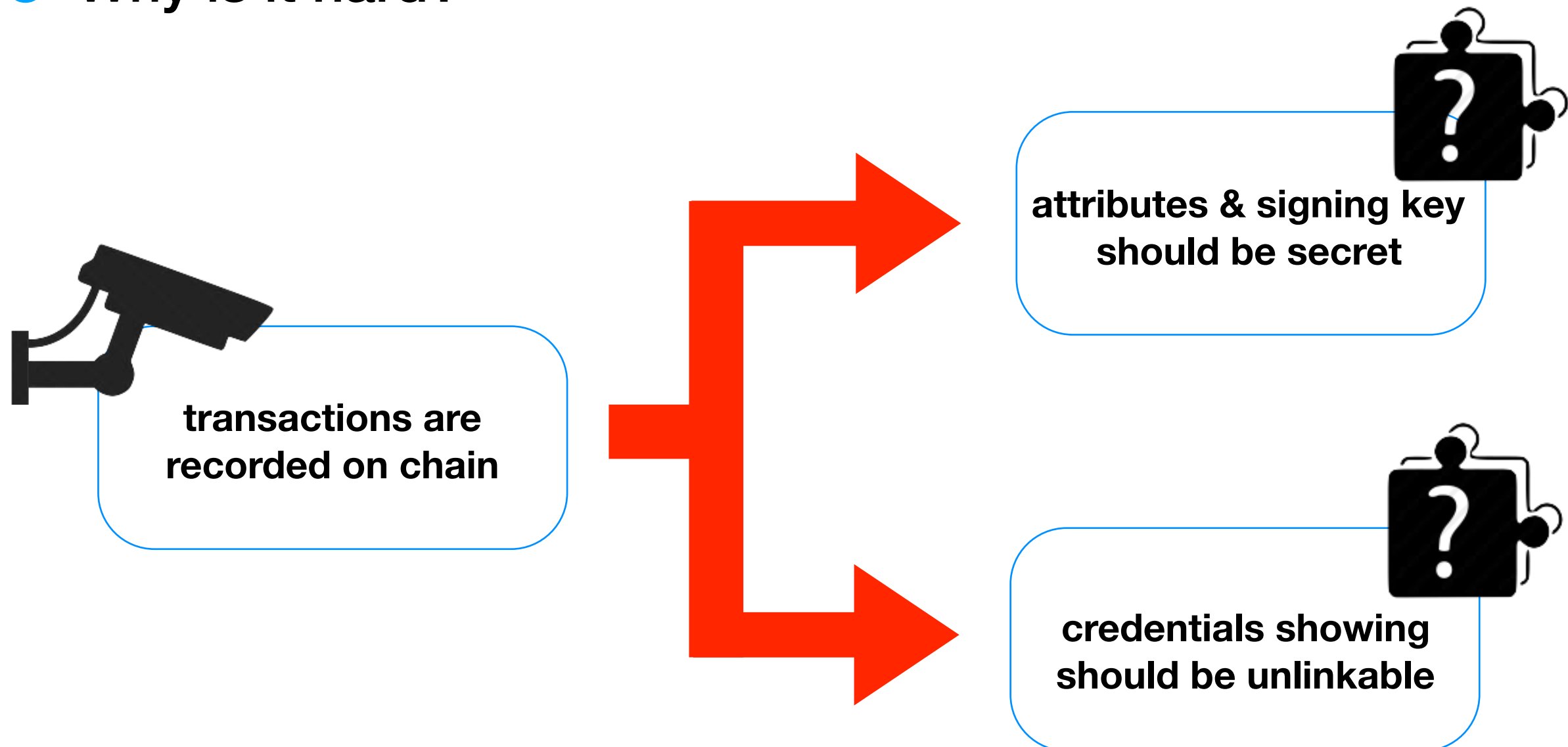
- Why is it hard?



**In a decentralised setting**

# What are we trying to do?

- Why is it hard?



**In a decentralised setting**

# Introduction

- Which properties do we need?

# Introduction

- Which properties do we need?

Blindness



# Introduction

- Which properties do we need?

Blindness



Unlinkability



# Introduction

- Which properties do we need?

Blindness



Unlinkability



Threshold Authority





# Introduction

- Which properties do we need?

Blindness



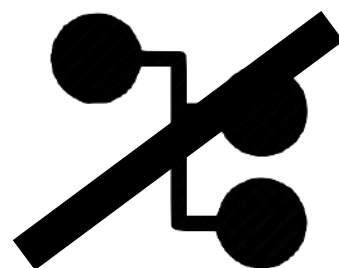
Unlinkability



Threshold Authority



Authorities Non-Interactivity



# Introduction

- Which properties do we need?

Blindness



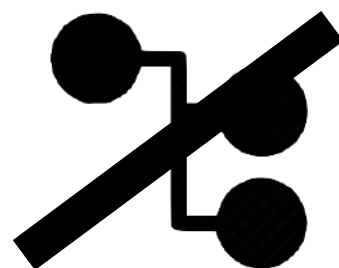
Unlinkability



Threshold Authority



Authorities Non-Interactivity



Efficiency



# So we built Coconut



# Introduction

- What is Coconut?

# Introduction

- What is Coconut?

## Contribution I

**Coconut credentials scheme**



# Introduction

- What is Coconut?

## Contribution I

**Coconut credentials scheme**



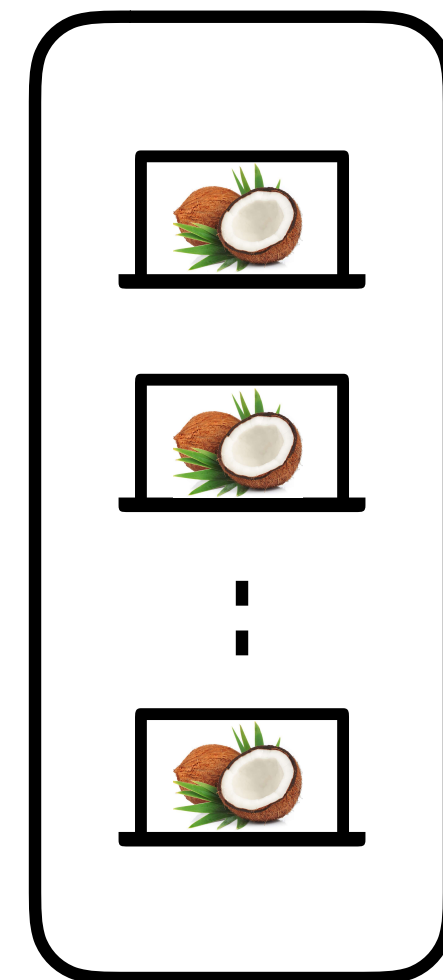
## Contribution II

**Coconut smart contract library & example of applications**



# System Overview

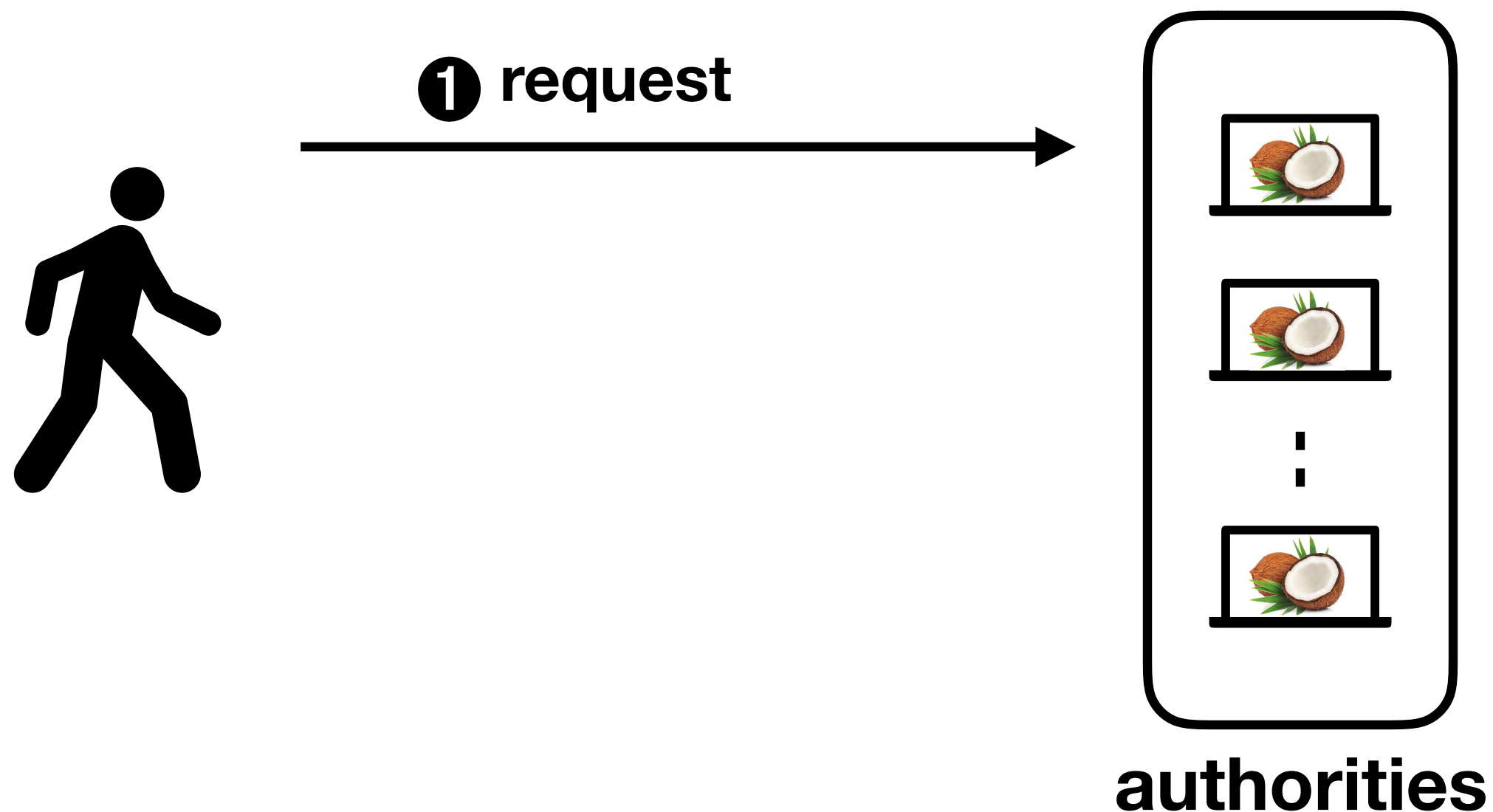
- How does Coconut work?



**authorities**

# System Overview

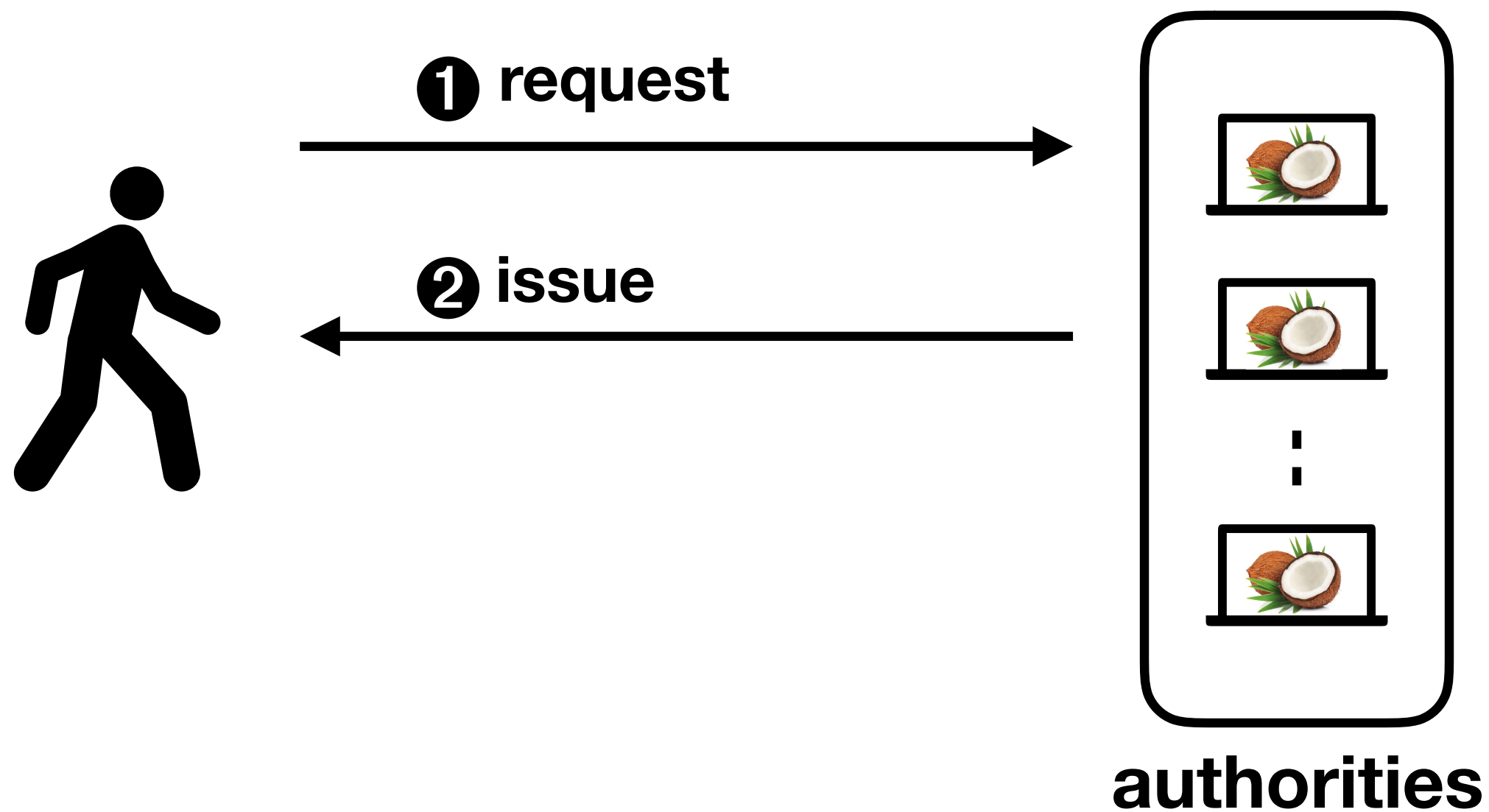
- How does Coconut work?





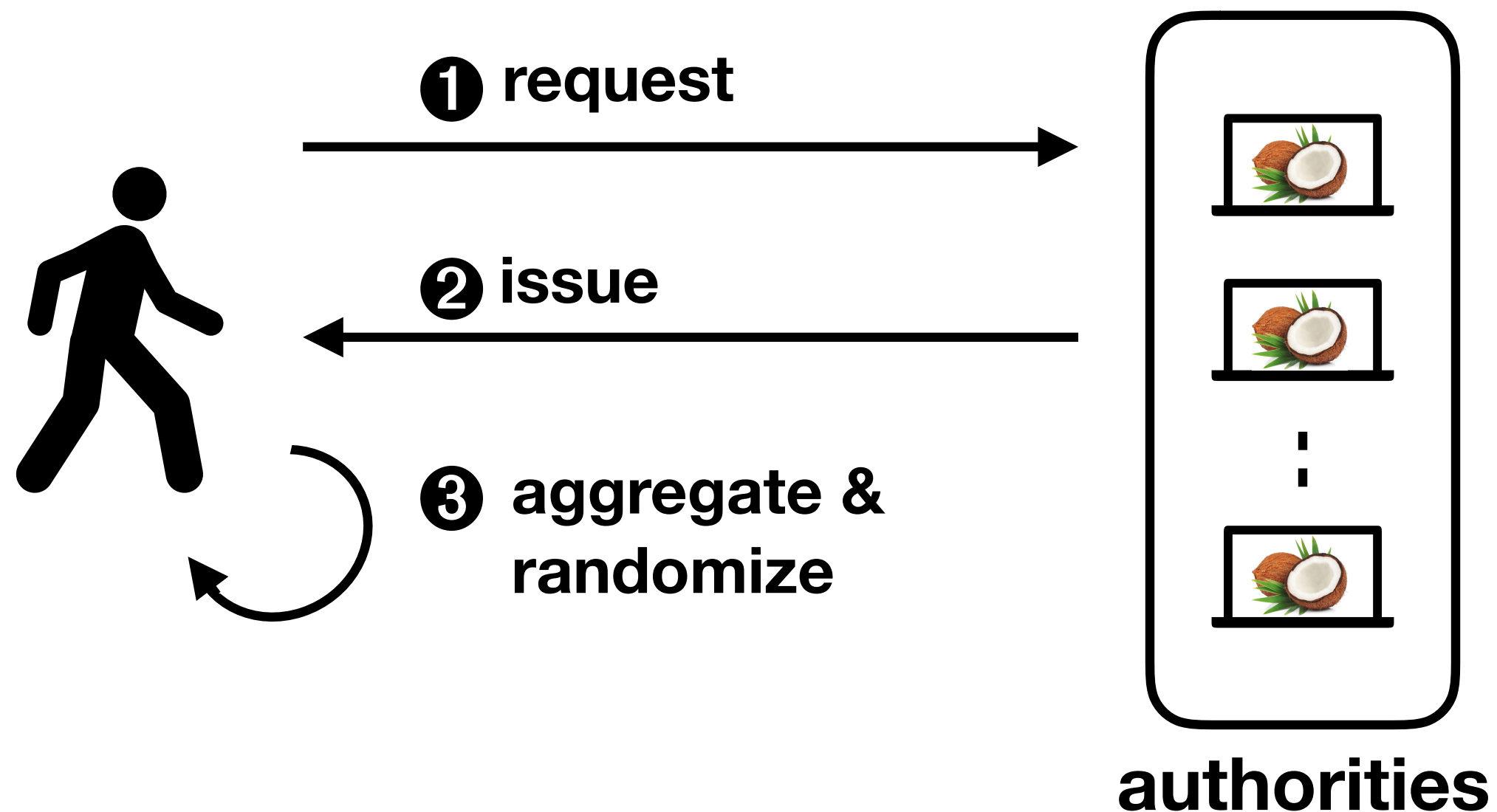
# System Overview

- How does Coconut work?



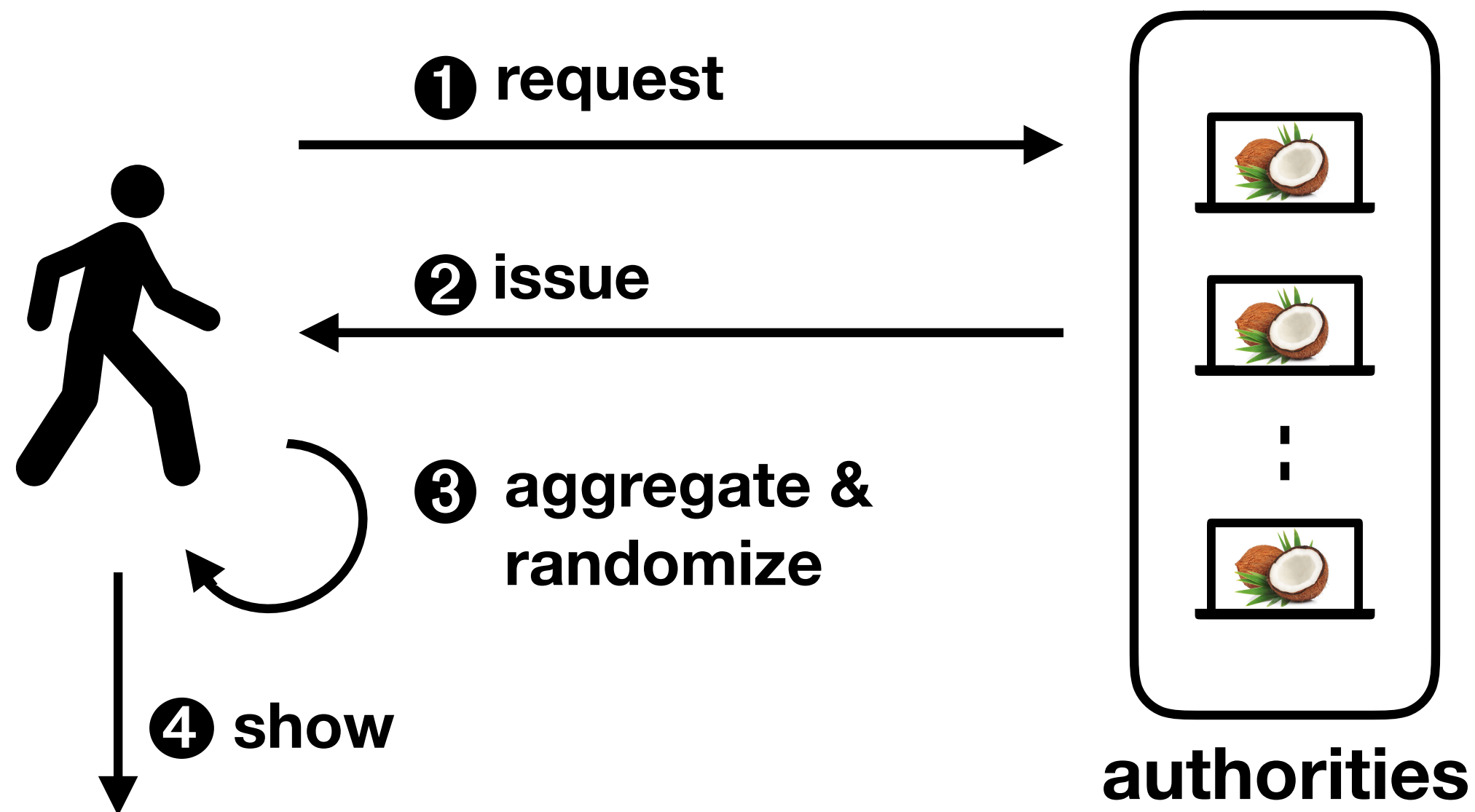
# System Overview

- How does Coconut work?



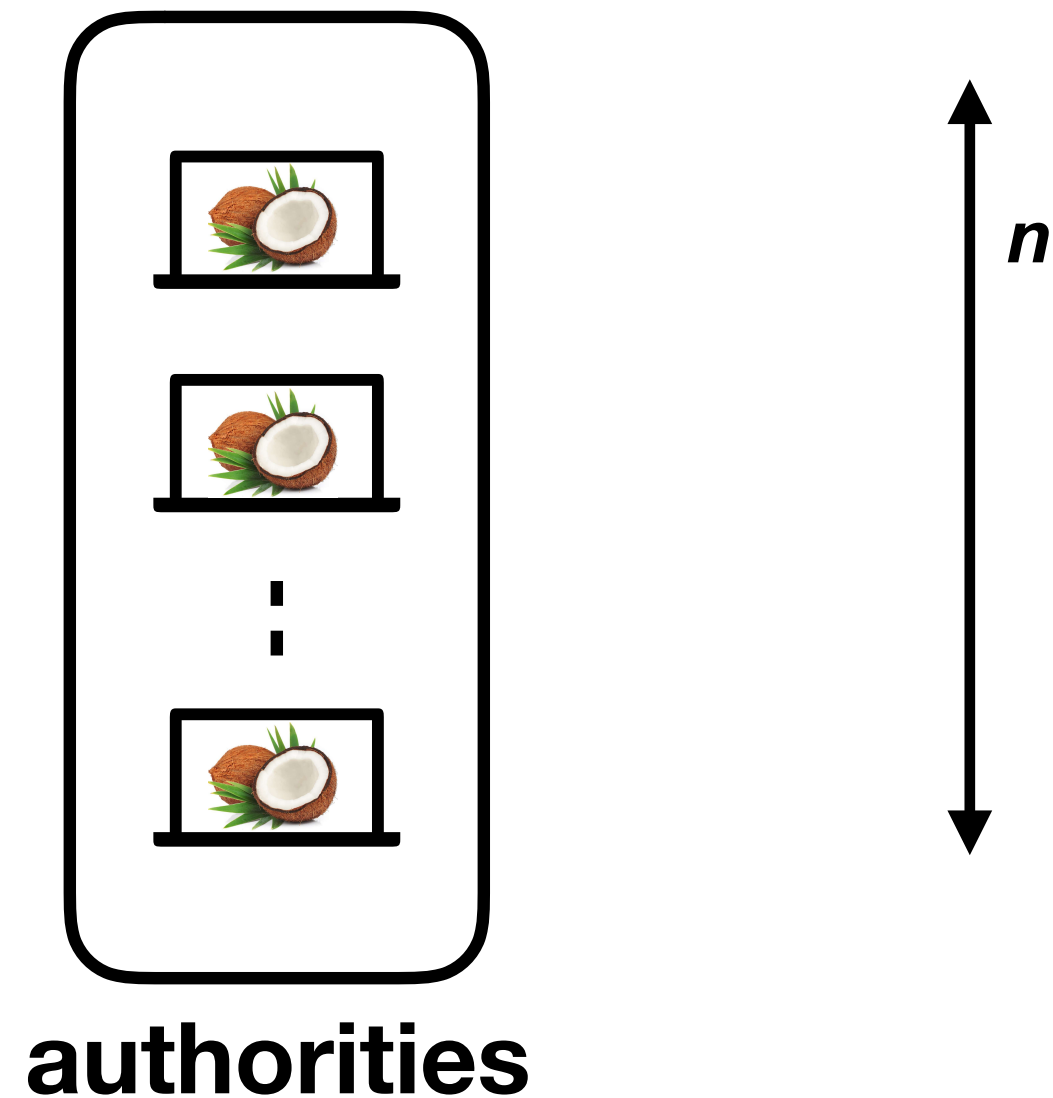
# System Overview

- How does Coconut work?



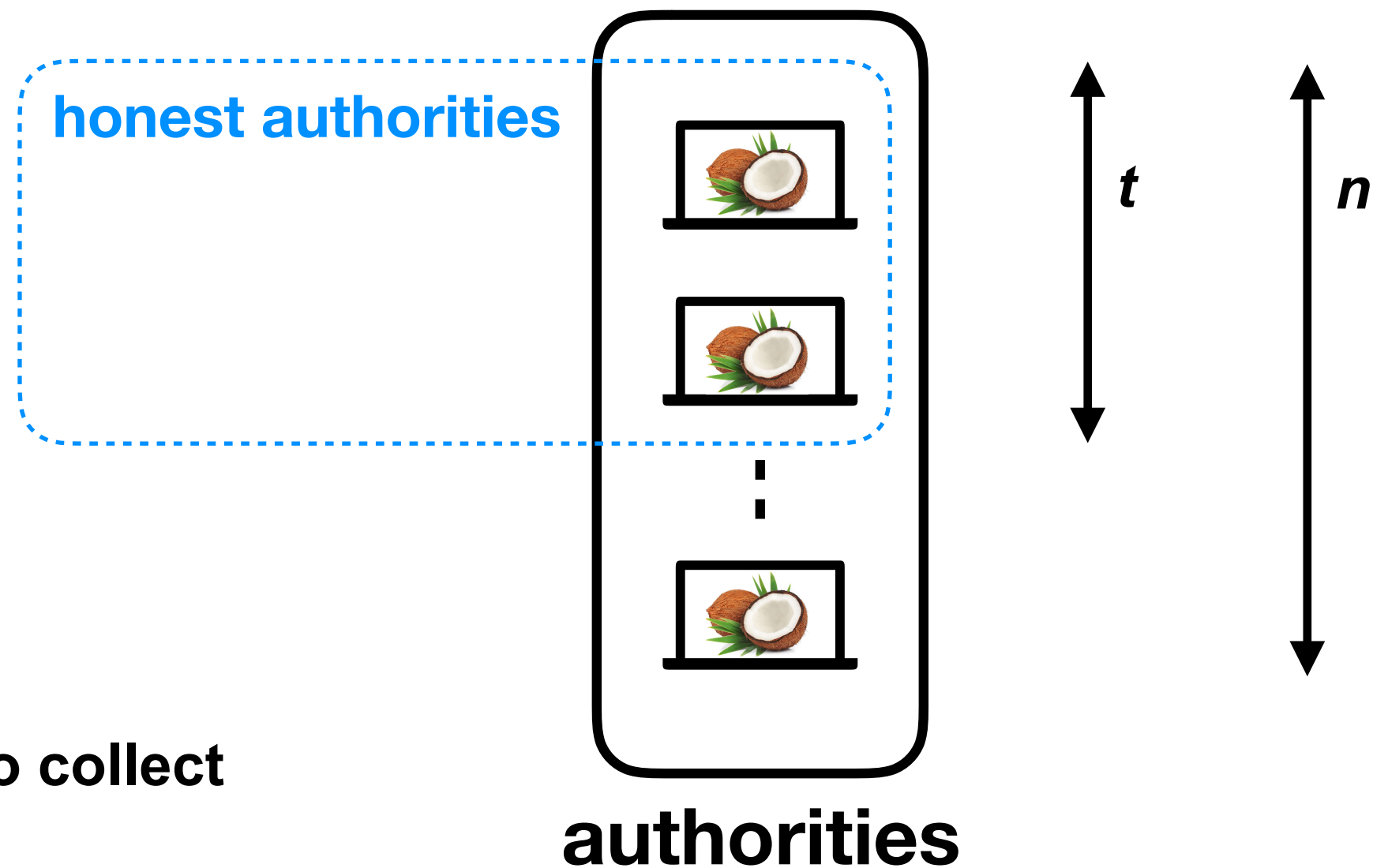
# System Overview

- Threshold authorities



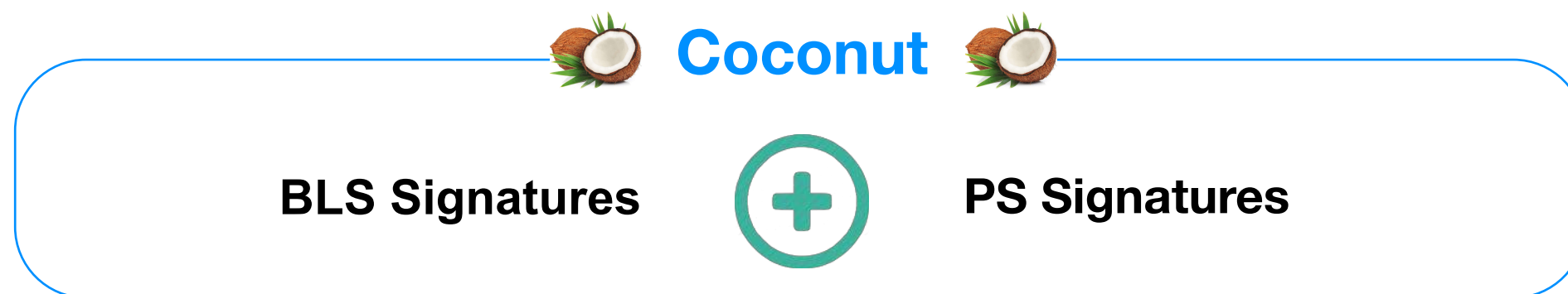
# System Overview

- Threshold authorities



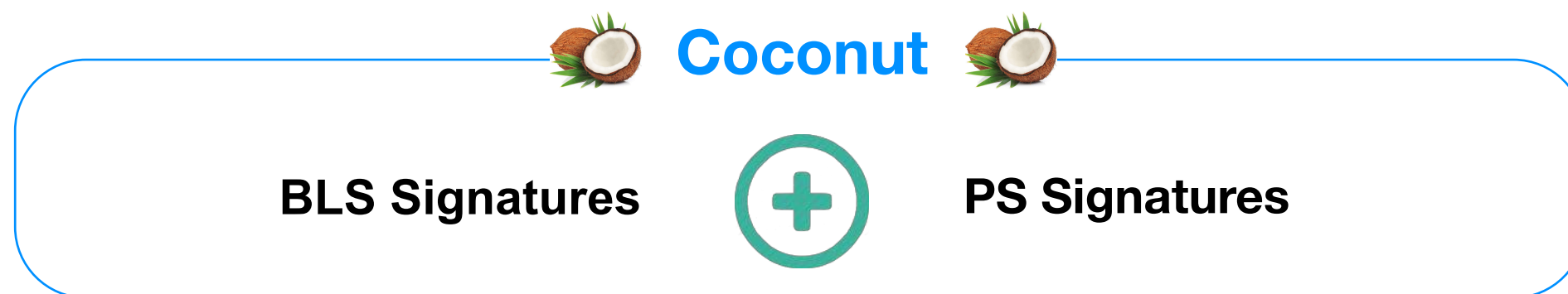
# Coconut Credentials Scheme

- From where do coconuts come from?



# Coconut Credentials Scheme

- From where do coconuts come from?



- What do they look like?

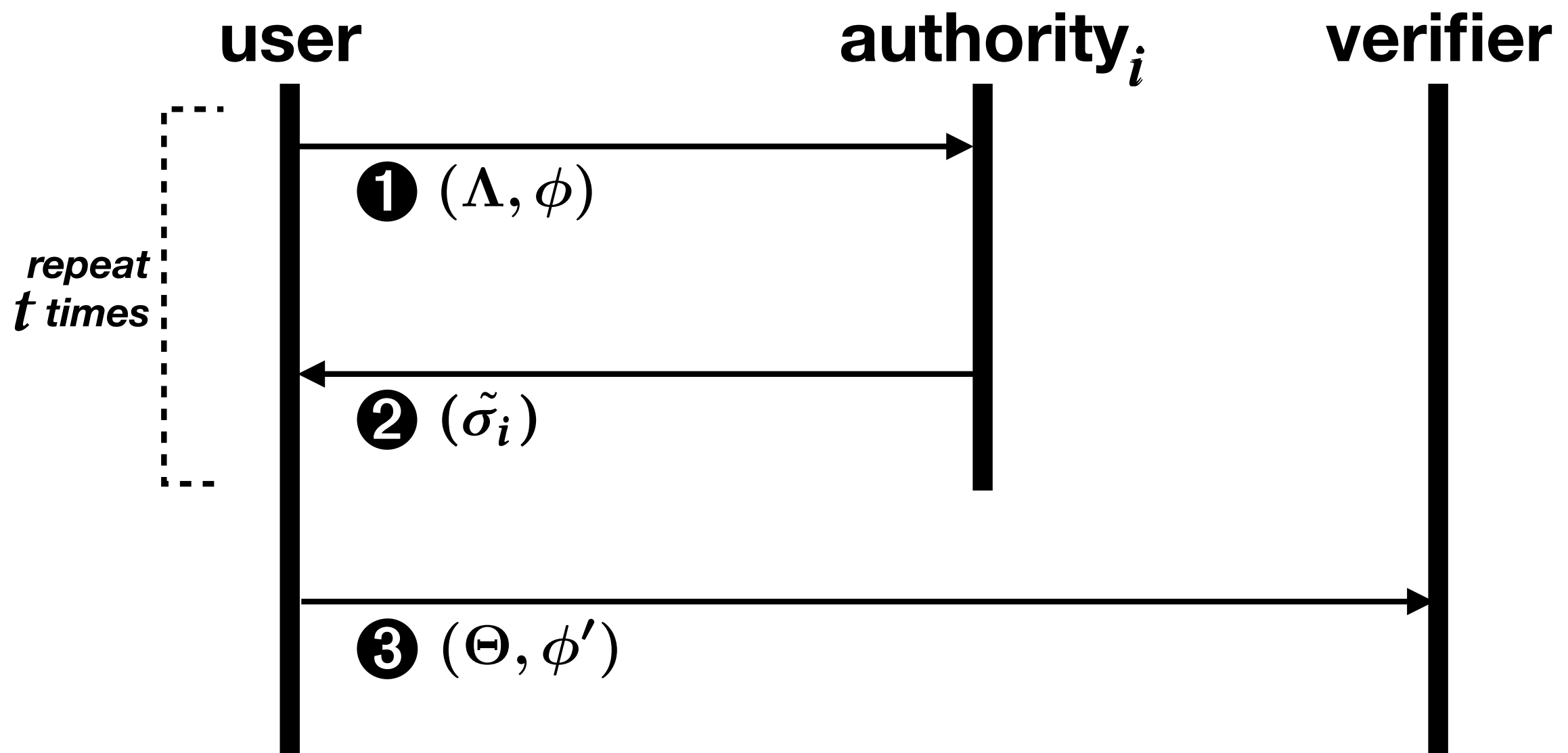
take an attribute:  $m$

compute:  $h \leftarrow H(c_m)$

signature:  $\sigma \leftarrow (h, h^{x+my})$  & secret key:  $(x, y)$

# Coconut Credentials Scheme

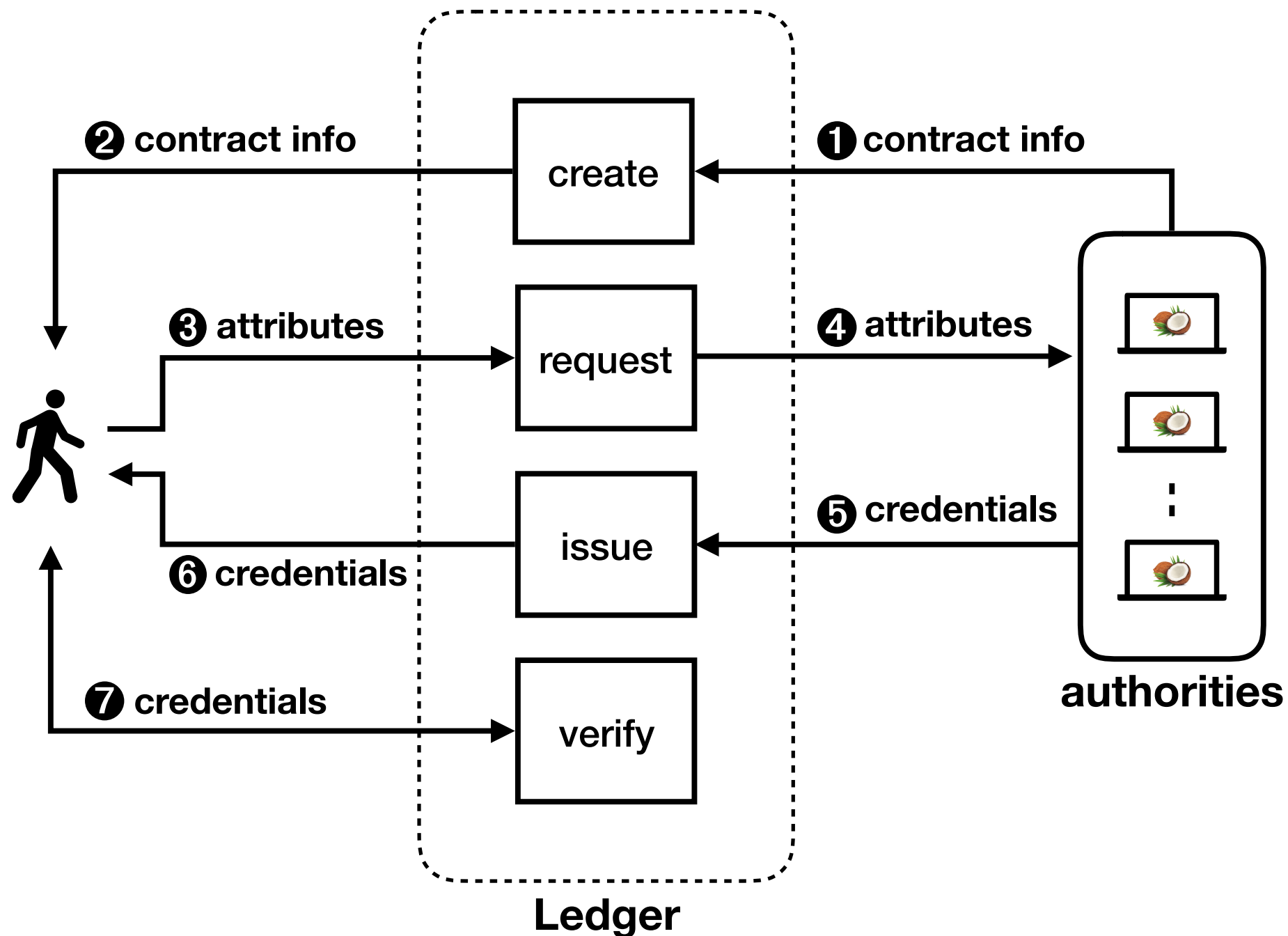
- Communication protocol





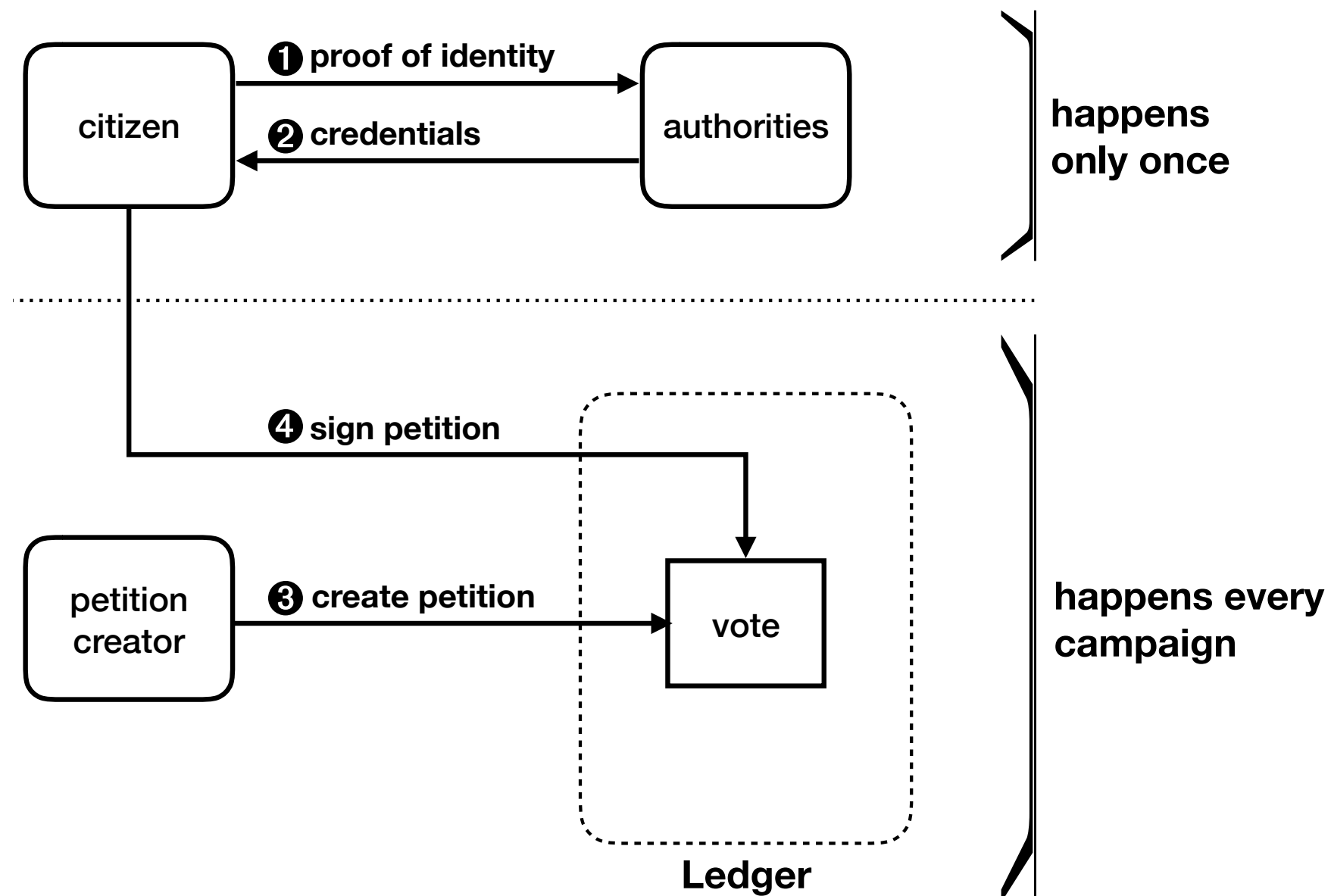
# Coconut Smart Contract Library

- General purpose library



# Applications

- Privacy-preserving petitions



# Performance

- What is out there?



# Performance

- What is out there?

**The Coconut  
cryptographic library**

**Python & Timing  
benchmark**



# Performance

- What is out there?

**The Coconut  
cryptographic library**

**Python & Timing  
benchmark**



**Smart contract library**



&



# Performance

## ● What is out there?

**The Coconut  
cryptographic library**

**Python & Timing  
benchmark**



**Smart contract library**



&



**Applications**

*Coin tumbler  
E-Petition  
(CRD proxy distribution )*

# Performance

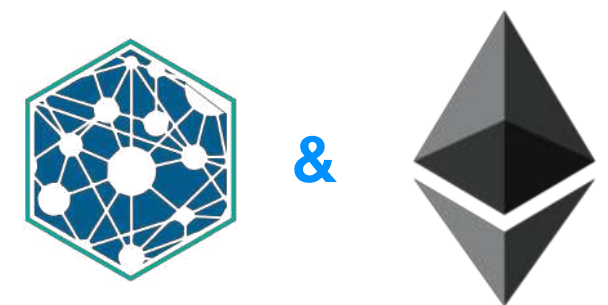
## ● What is out there?

**The Coconut  
cryptographic library**

**Python & Timing  
benchmark**



**Smart contract library**

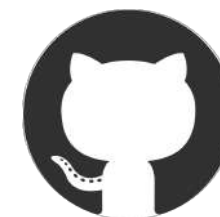


**Applications**

*Coin tumbler  
E-Petition  
(CRD proxy distribution )*

**Everything is released as open source software**

<https://github.com/asonnino/coconut>



# Performance

## ● How fast is Coconut?

	Operation	$\mu$ [ms]	$\sqrt{\sigma^2}$ [ms]
sign	PrepareBlindSign	2.633	$\pm 0.003$
	BlindSign	3.356	$\pm 0.002$
	Unblind	0.445	$\pm 0.002$
	AggCred	0.454	$\pm 0.000$
verify	ProveCred	1.544	$\pm 0.001$
	VerifyCred	10.497	$\pm 0.002$

signing is fast, verifying takes 10ms



# Performance

- What is the size of the credentials?

**2 Group Elements**

**No matter how many attributes...**

**No matter how many authorities...**

# Performance

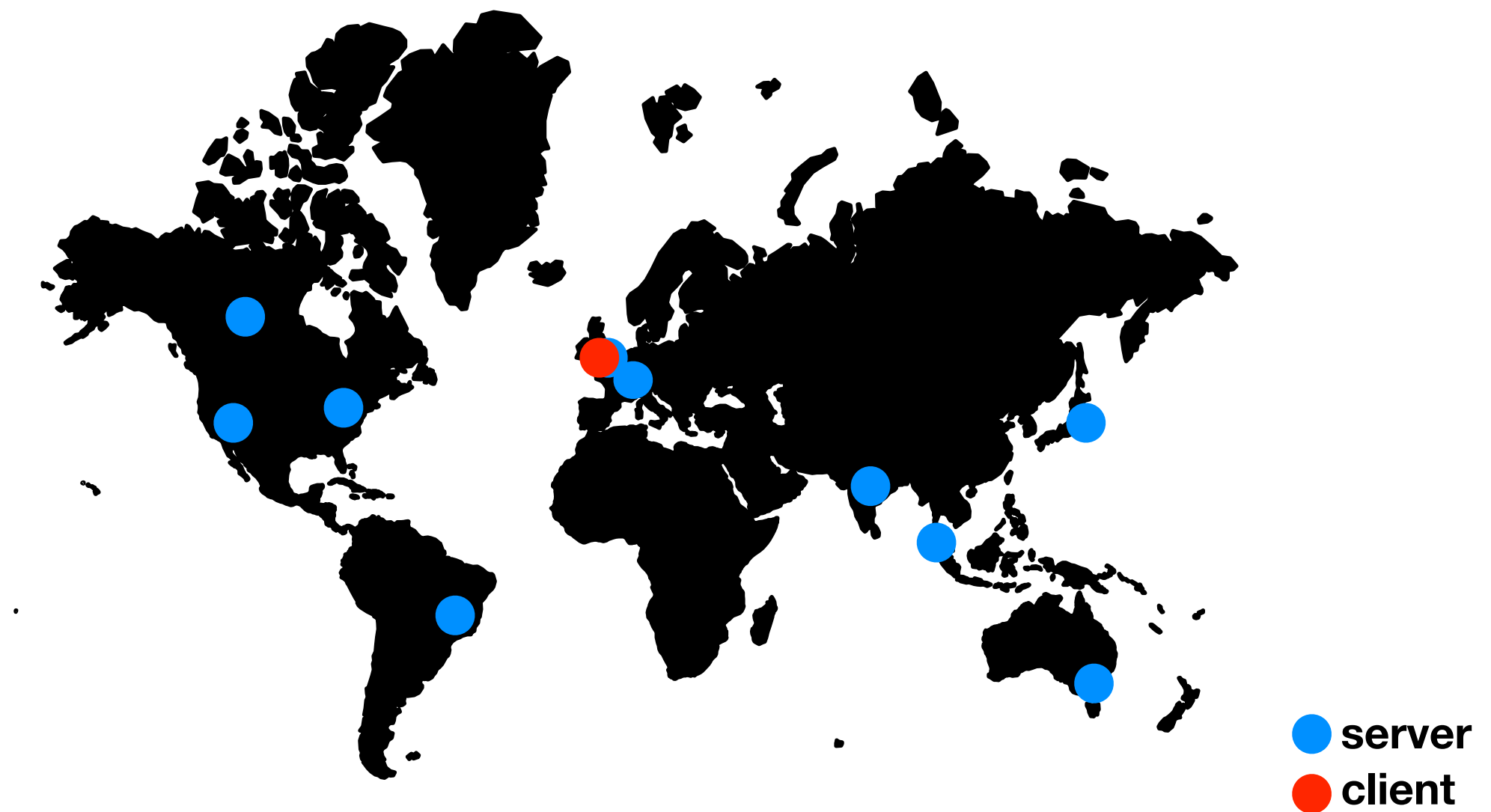
## ● How does Coconut scale?

Number of authorities: $n$ , Signature size: 132 bytes		
Transaction	complexity	size [B]
Signature on public attribute:		
① request credential	$O(n)$	32
② issue credential	$O(n)$	132
③ verify credential	$O(1)$	162
Signature on private attribute:		
issue ① request credential	$O(n)$	516
② issue credential	$O(n)$	132
verify ③ verify credential	$O(1)$	355

Signing scales linearly, verifying is constant time

# Performance

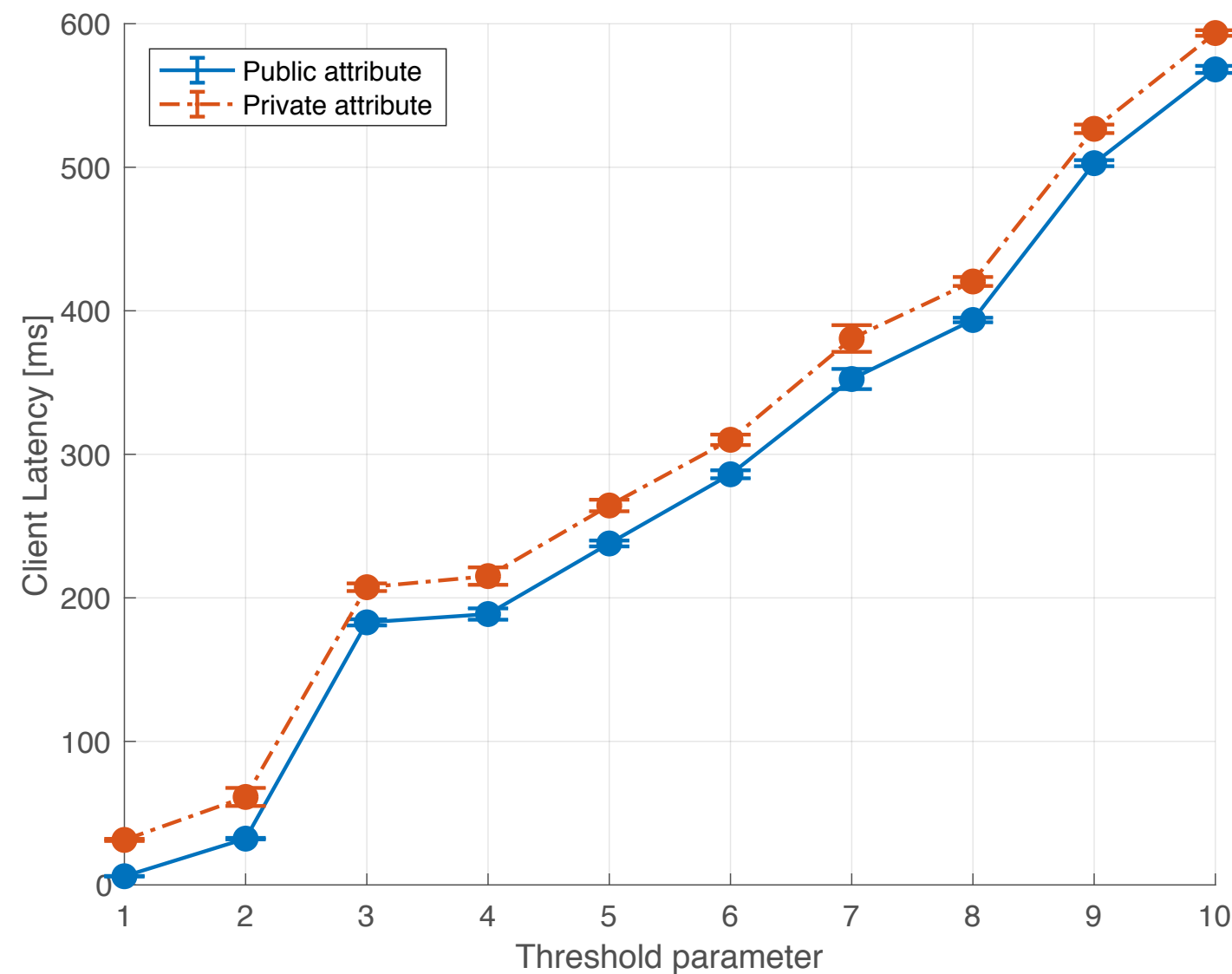
- Did you evaluate it in the real world?



pick 10 locations across the world

# Performance

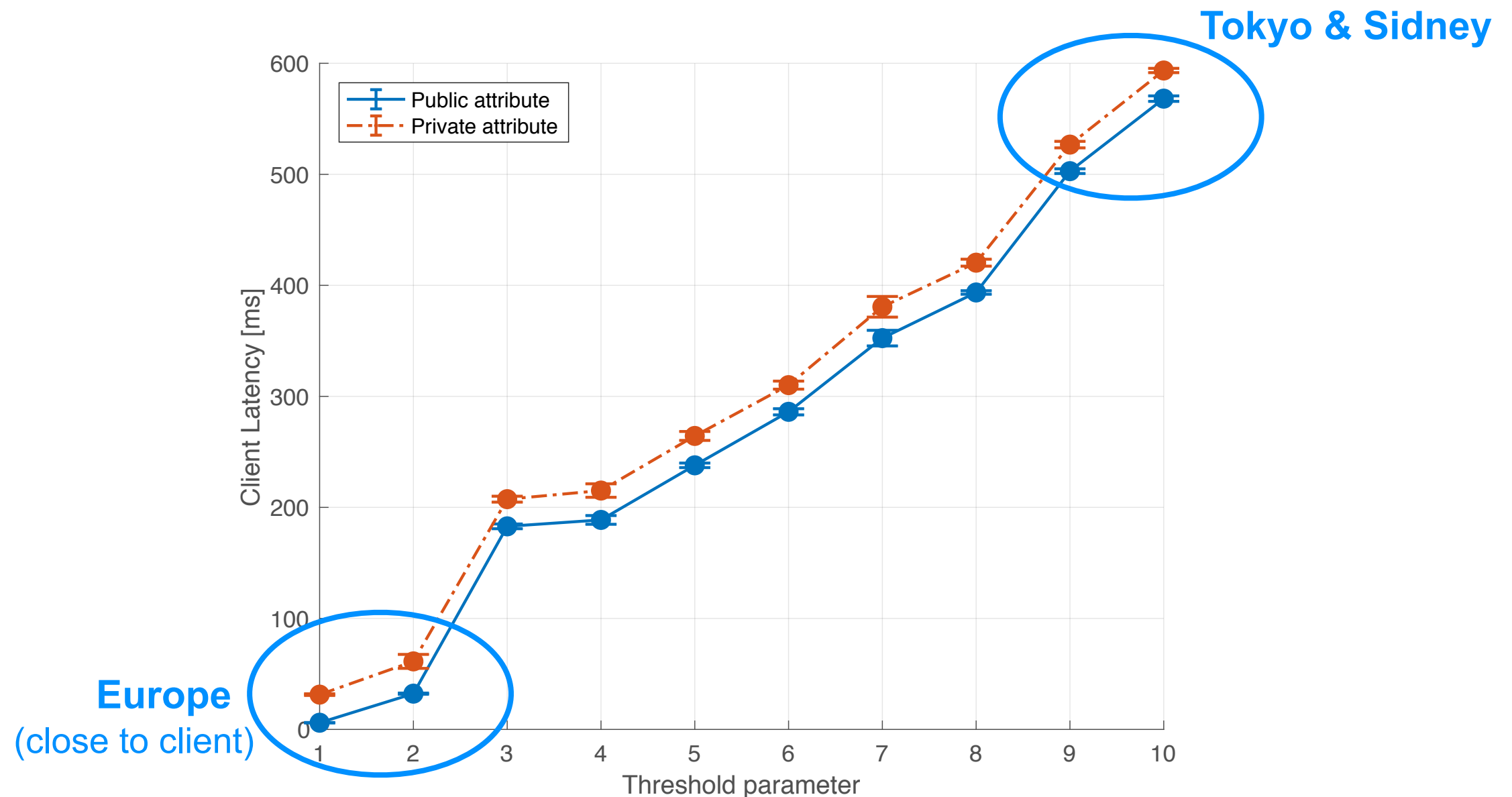
- Did you evaluate it in the real world?



client latency VS number of authorities

# Performance

- Did you evaluate it in the real world?



client latency VS number of authorities

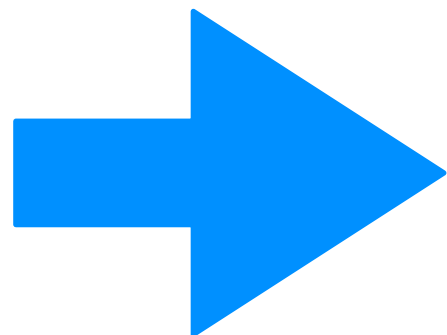
# What else is in the paper?

**Full cryptographic scheme**

**Smart contract library evaluation**

**Coin tumbler, CRD proxy applications**

**Applications evaluation and benchmarking**



arXiv:submit/2158644 [cs.CR] 20 Feb 2018

## Coconut: Threshold Issuance Selective Disclosure Credentials with Applications to Distributed Ledgers

Alberto Sonnino  
University College London

Mustafa Al-Bassam  
University College London

Shehar Bano  
University College London

George Danezis  
University College London  
The Alan Turing Institute

### Abstract

We present Coconut, a novel selective disclosure credential scheme supporting distributed threshold issuance, public and private attributes, re-randomization, and multiple unlinkable selective attribute revelations. Coconut can be used by modern blockchains to ensure confidentiality, authenticity and availability even when a subset of credential issuing authorities are malicious or offline. We implement and evaluate a generic Coconut smart contract library for Chainspace and Ethereum; and present three applications related to anonymous payments, electronic petitions, and distribution of proxies for censorship resistance. Coconut uses short and computationally efficient credentials, and our evaluation shows that most Coconut cryptographic primitives take just a few milliseconds on average, with verification taking the longest time (10 milliseconds).

### 1 Introduction

Selective disclosure credentials [15, 17] allow the issuance of a credential to a user, and the subsequent unlinkable revelation (or ‘showing’) of some of the attributes it encodes to a verifier for the purposes of authentication, authorization or to implement electronic cash. However, established schemes have shortcomings. Some entrust a single issuer with the credential signature key, allowing a malicious issuer to forge any credential or electronic coin. Other schemes do not provide the necessary re-randomization or blind issuing properties necessary to implement modern selective disclosure credentials. No existing scheme provides all of threshold distributed issuance, private attributes, re-randomization, and unlinkable multi-show selective disclosure.

The lack of full-featured selective disclosure credentials impacts platforms that support ‘smart contracts’, such as Ethereum [40], Hyperledger [14] and Chainspace [3]. They all share the limitation that ver-

ifiable smart contracts may only perform operations recorded on a public blockchain. Moreover, the security models of these systems generally assume that integrity should hold in the presence of a threshold number of dishonest or faulty nodes (Byzantine fault tolerance); it is desirable for similar assumptions to hold for multiple credential issuers (threshold aggregability).

Issuing credentials through smart contracts would be very desirable: a smart contract could conditionally issue user credentials depending on the state of the blockchain, or attest some claim about a user operating through the contract—such as their identity, attributes, or even the balance of their wallet. This is not possible, with current selective credential schemes that would either entrust a single party as an issuer, or would not provide appropriate re-randomization, blind issuance and selective disclosure capabilities (as in the case of threshold signatures [5]). For example, the Hyperledger system supports CL credentials [15] through a trusted third party issuer, illustrating their usefulness, but also their fragility against the issuer becoming malicious.

Coconut addresses this challenge, and allows a subset of decentralized mutually distrustful authorities to jointly issue credentials, on public or private attributes. Those credentials cannot be forged by users, or any small subset of potentially corrupt authorities. Credentials can be re-randomized before selected attributes being shown to a verifier, protecting privacy even in the case all authorities and verifiers collude. The Coconut scheme is based on a threshold issuance signature scheme, that allows partial claims to be aggregated into a single credential. Mapped to the context of permissioned and semi-permissioned blockchains, Coconut allows collections of authorities in charge of maintaining a blockchain, or a side chain [5] based on a federated peg, to jointly issue selective disclosure credentials.

Coconut uses short and computationally efficient credentials, and efficient revelation of selected attributes and verification protocols. Each partial credentials and the

# Limitations & Future Works

- Would you like to contribute?

## Limitation I

**Adding and removing authorities is complicated.  
Can we do better than re-running the key generation algorithm?**

# Limitations & Future Works

- Would you like to contribute?

## Limitation I

**Adding and removing authorities is complicated.  
Can we do better than re-running the key generation algorithm?**

## Limitation II

**Current key generation algorithms are complex to implement.  
Can we design a key generation algorithm for blockchains?**



# Limitations & Future Works

- What is the next milestone?

**A general framework allowing nodes to execute any kind of threshold cryptography?**

# Conclusion

- What did we talk about?

## Contribution I

**Coconut credentials scheme**



## Contribution II

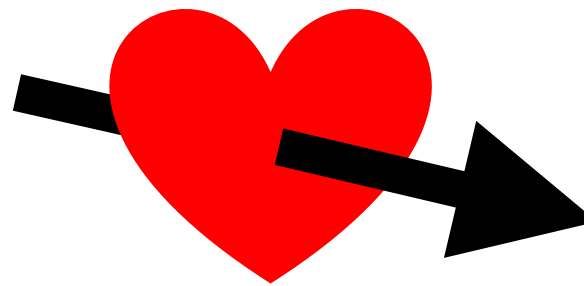
**Coconut smart contract library & example of applications**



# Conclusion

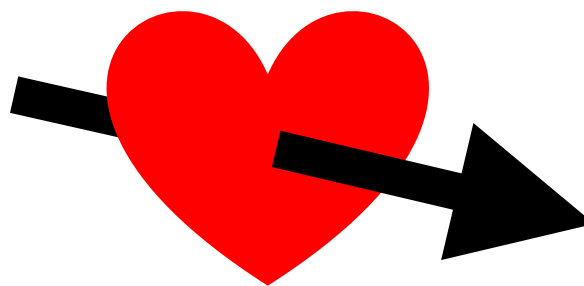
- Main take-aways

**Threshold  
issuance**



**Sweet for  
blockchains**

**Randomizable**



**Multi-use &  
unlinkability**

**Thank you for your attention**

---

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



<https://github.com/asonnino/coconut>