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

write the contract → send it to the blockchain → anyone can verify
Challenges in blockchains

Can we issue credentials in this setting?

- write the contract
- send it to the blockchain
- anyone can verify
What are we trying to do?

- Issuing credentials through smart contracts

... while preserving privacy
What are we trying to do?

- Issuing credentials through smart contracts

... 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

... 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:
- transactions are recorded on chain
- attributes & signing key should be secret
What are we trying to do?

- Why is it hard?

In a decentralised setting:

- Transactions are recorded on chain
- Attributes & signing key should be secret
- Credentials showing should be unlinkable
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?
System Overview

How does Coconut work?

1. request

authorities
System Overview

- How does Coconut work?

1. request
2. issue

authorities
System Overview

How does Coconut work?

1. request
2. issue
3. aggregate & randomize
System Overview

- How does Coconut work?

1. Request
2. Issue
3. Aggregate & randomize
4. Show

authorities
System Overview

- Threshold authorities
System Overview

- Threshold authorities

Users need to collect only $t$ shares
Coconut Credentials Scheme

- From where do coconuts come from?

Coconut

BLS Signatures

PS Signatures
Coconut Credentials Scheme

- From where do coconuts come from?

![Diagram of Coconut]

- What do they look like?

  take an attribute: \( m \)
  
  compute: \( h \leftarrow H(c_m) \)
  
  signature: \( \sigma \leftarrow (h, h^{x+my}) \) & secret key: \( (x, y) \)
This appendix sketches the security proofs of the cryptographic construction described in Section 3.

Blindness.

Unforgeability of the Malet library contract and a similar system.

The unforgeability of the Malet library contract and a similar system.

The unlinkability property means that the verifier, exactly as intended.

Adding one embeds the users compute a value.

The signature scheme has been implemented in python.

Table I shows the mean of CoCoNut when deeply related to blockchains), since we also prevent a single authority from taking the user wish to issue some long term credentials to their authorities issuing the credentials. In order to sign the petition, they add their vote to the options, append their signature, and are expecting.

Adding a custom function requiring the user to prove in some ways that signing is much faster than verifying signatures (about an order of magnitude faster).

The complexity is expressed as the number of measures: 10,000

TABLE II: Communication complexity and transaction size.

<table>
<thead>
<tr>
<th></th>
<th>Communication complexity</th>
<th>Transaction size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signature</td>
<td>132 bytes</td>
<td>132 bytes</td>
</tr>
<tr>
<td>Number of measures</td>
<td>10,000</td>
<td>10,000</td>
</tr>
<tr>
<td>Signature size</td>
<td>132 bytes</td>
<td>132 bytes</td>
</tr>
<tr>
<td>Number of repeats</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Alberto: Update the above

Alberto: compare results (speed and size) with alternatives

TABLE II: Communication complexity and transaction size.

<table>
<thead>
<tr>
<th></th>
<th>Communication complexity</th>
<th>Transaction size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signature</td>
<td>132 bytes</td>
<td>132 bytes</td>
</tr>
<tr>
<td>Number of measures</td>
<td>10,000</td>
<td>10,000</td>
</tr>
<tr>
<td>Signature size</td>
<td>132 bytes</td>
<td>132 bytes</td>
</tr>
<tr>
<td>Number of repeats</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Coconut Smart Contract Library

- General purpose library

![Diagram](image_url)
Applications

- Privacy-preserving petitions

---

1. proof of identity
2. credentials
3. create petition
4. sign petition

---

happens only once

happens every campaign
Performance

- What is out there?

https://github.com/asonnino/coconut
Everything is released as open source software
Performance

- What is out there?

The Coconut cryptographic library

Python & Timing benchmark

https://github.com/asonnino/coconut

Everything is released as open source software
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

Applications

Coin tumbler
E-Petition
(CRD proxy distribution)

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)

Everything is released as open source software

https://github.com/asonnino/coconut
Performance

How fast is Coconut?

<table>
<thead>
<tr>
<th>Operation</th>
<th>( \mu ) [ms]</th>
<th>( \sqrt{\sigma^2} ) [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrepareBlindSign</td>
<td>2.633</td>
<td>( \pm 0.003 )</td>
</tr>
<tr>
<td>BlindSign</td>
<td>3.356</td>
<td>( \pm 0.002 )</td>
</tr>
<tr>
<td>Unblind</td>
<td>0.445</td>
<td>( \pm 0.002 )</td>
</tr>
<tr>
<td>AggCred</td>
<td>0.454</td>
<td>( \pm 0.000 )</td>
</tr>
<tr>
<td>ProveCred</td>
<td>1.544</td>
<td>( \pm 0.001 )</td>
</tr>
<tr>
<td>VerifyCred</td>
<td>10.497</td>
<td>( \pm 0.002 )</td>
</tr>
</tbody>
</table>

sign

verify

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

<table>
<thead>
<tr>
<th>Transaction</th>
<th>complexity</th>
<th>size [B]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signature on public attribute:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 request credential</td>
<td>( O(n) )</td>
<td>32</td>
</tr>
<tr>
<td>2 issue credential</td>
<td>( O(n) )</td>
<td>132</td>
</tr>
<tr>
<td>3 verify credential</td>
<td>( O(1) )</td>
<td>162</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Signature on private attribute:</th>
</tr>
</thead>
<tbody>
<tr>
<td>issue</td>
</tr>
<tr>
<td>1 request credential</td>
</tr>
<tr>
<td>2 issue credential</td>
</tr>
<tr>
<td>verify</td>
</tr>
<tr>
<td>3 verify credential</td>
</tr>
</tbody>
</table>

Signing scales linearly, verifying is constant time
Performance

Did you evaluate it in the real world?

pick 10 locations across the world
Did you evaluate it in the real world?

Performance

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
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

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.