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

November 2018

The Authors



Alberto Sonnino



Mustafa Al-Bassam



Bano Shehar



Sarah Meiklejohn



George Danezis

Alberto Sonnino

- Ph.D researcher UCL
- Co-founder of Chainspace

Background:



University College London (UCL)

Master Degree — Information Security United Kingdom



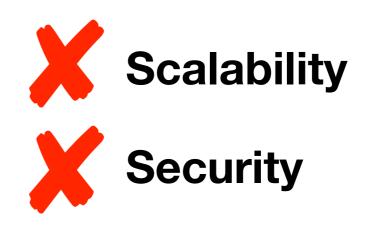
Karlsruhe Institute of Technology (KIT) Master Degree — Information Technology Engineering Germany



Catholic University of Louvain (UCLouvain) Master Degree — Information Technology Engineering Belgium

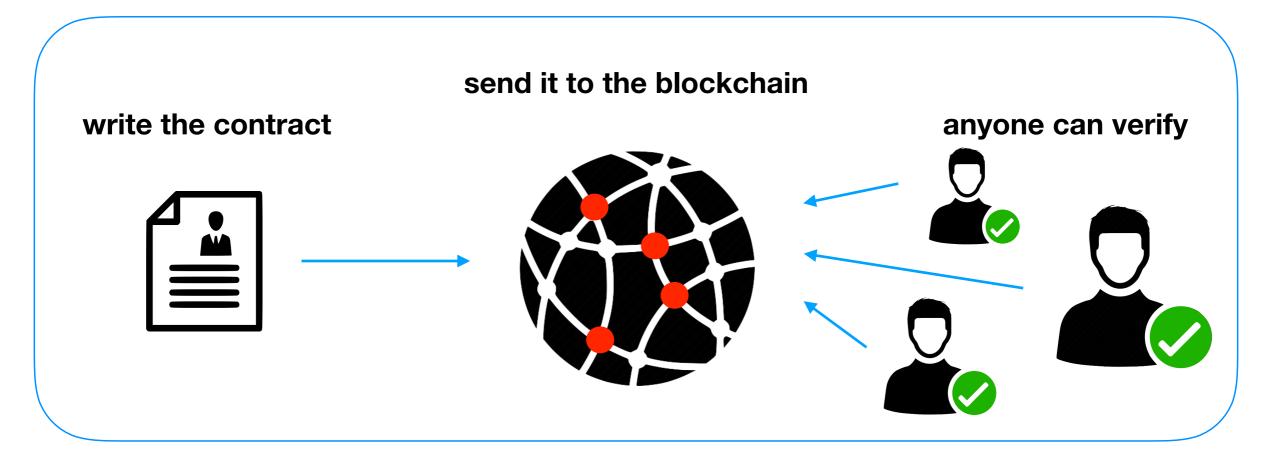
Challenges in blockchains





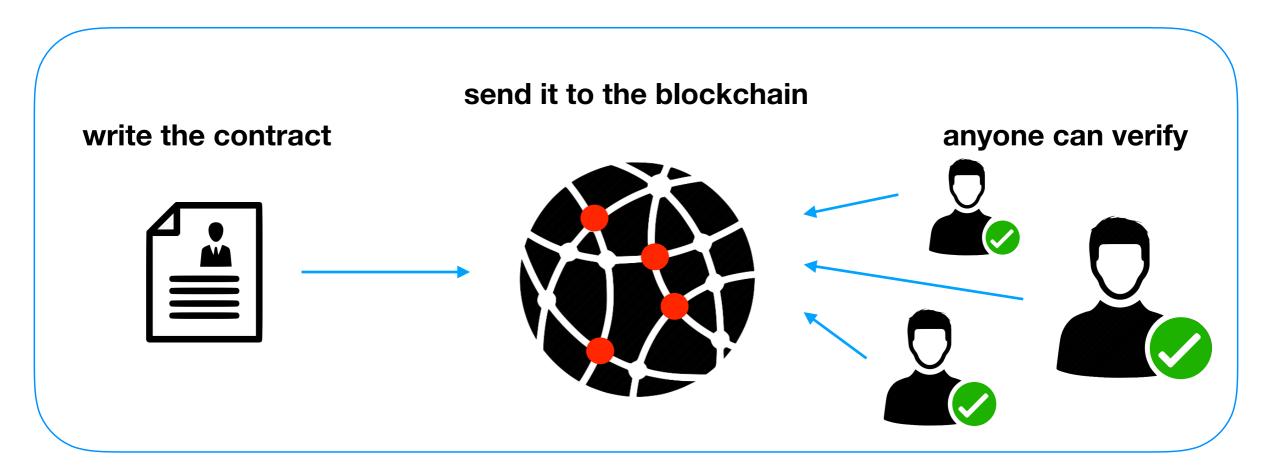
Challenges in blockchains





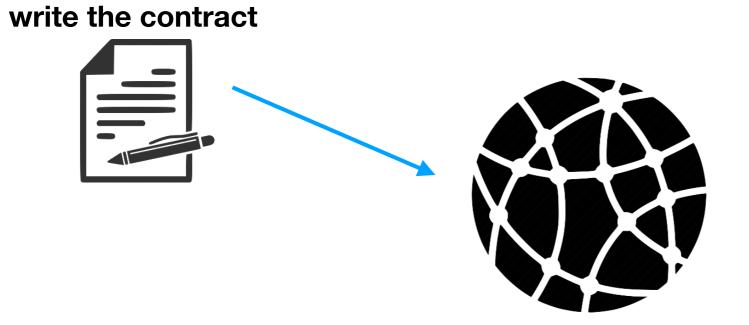
Challenges in blockchains

Can we issue credentials in this setting?



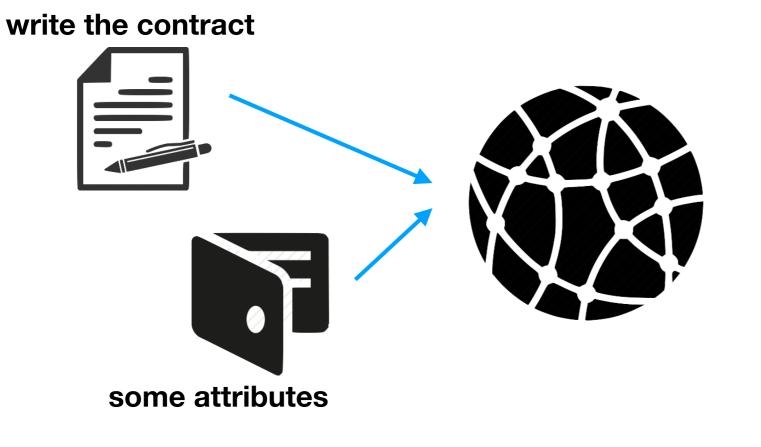
What are we trying to do?

Issuing credentials through smart contracts



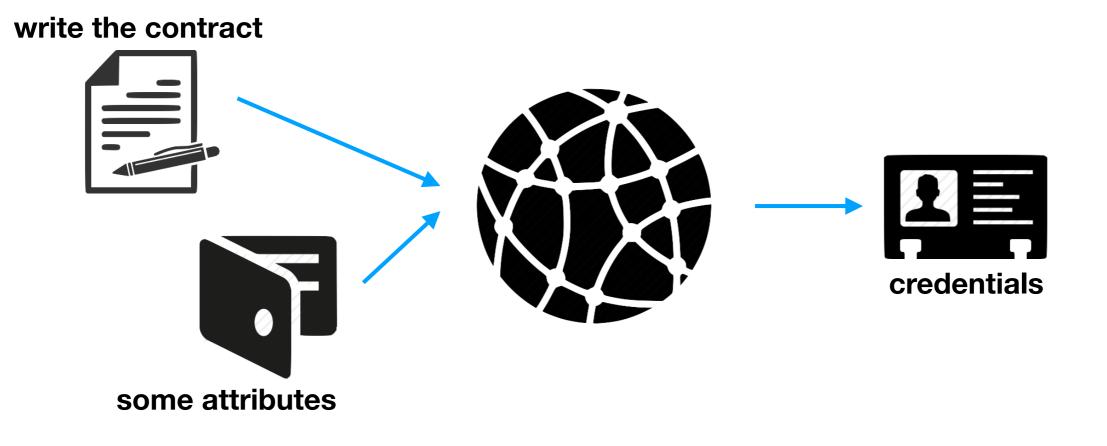
What are we trying to do?

Issuing credentials through smart contracts



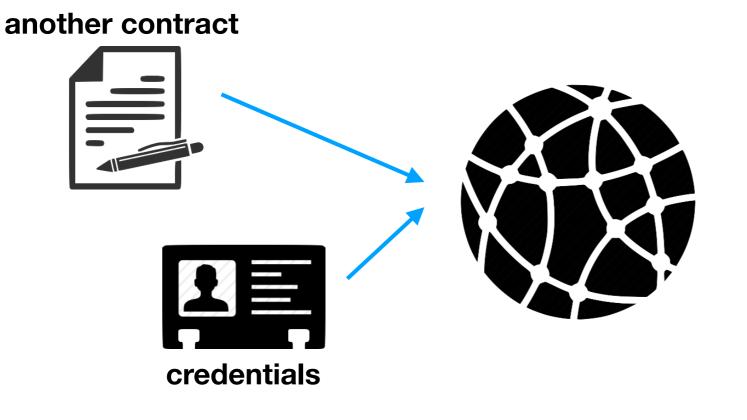
What are we trying to do?

Issuing credentials through smart contracts



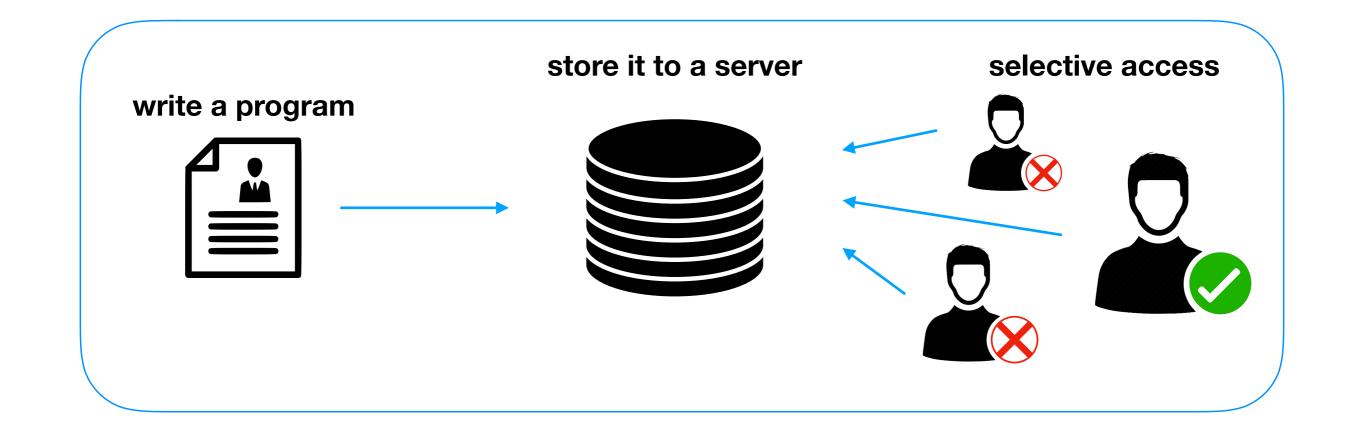
What are we trying to do?

Issuing credentials through smart contracts



What are we trying to do?

The more traditional setting



... but without any central authority

What are we trying to do?

Why is it hard?

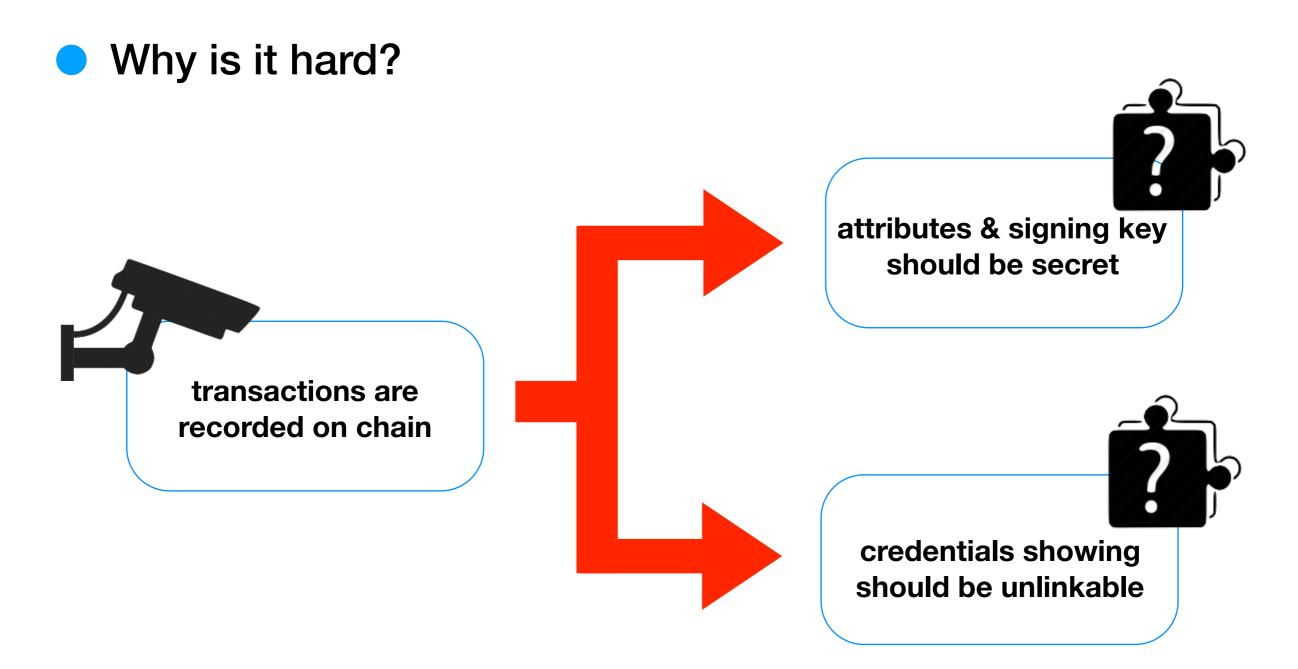


In a decentralised setting

What are we trying to do?

In a decentralised setting

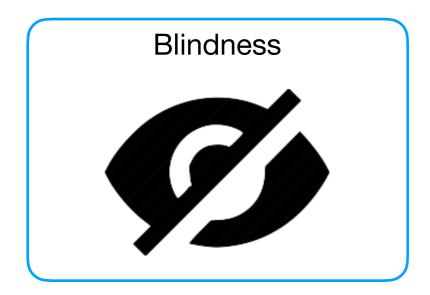
What are we trying to do?



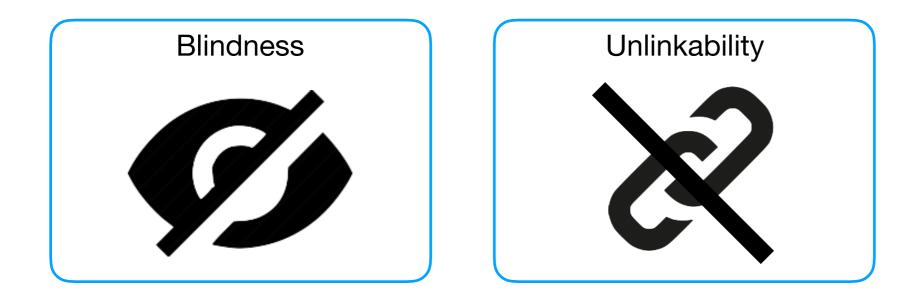
In a decentralised setting

Introduction

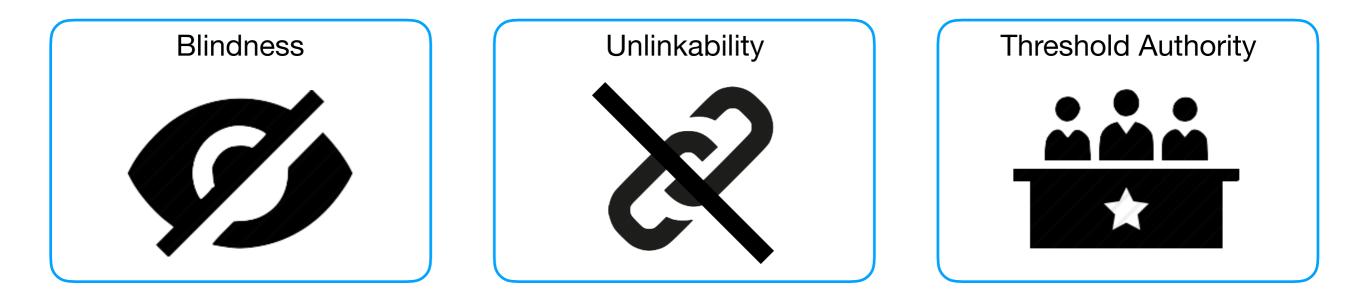
Introduction



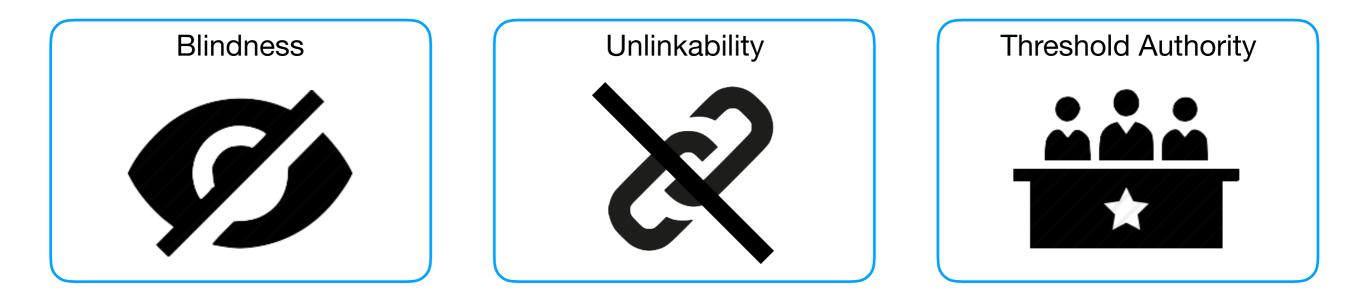
Introduction

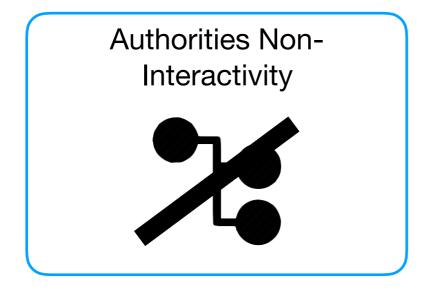


Introduction

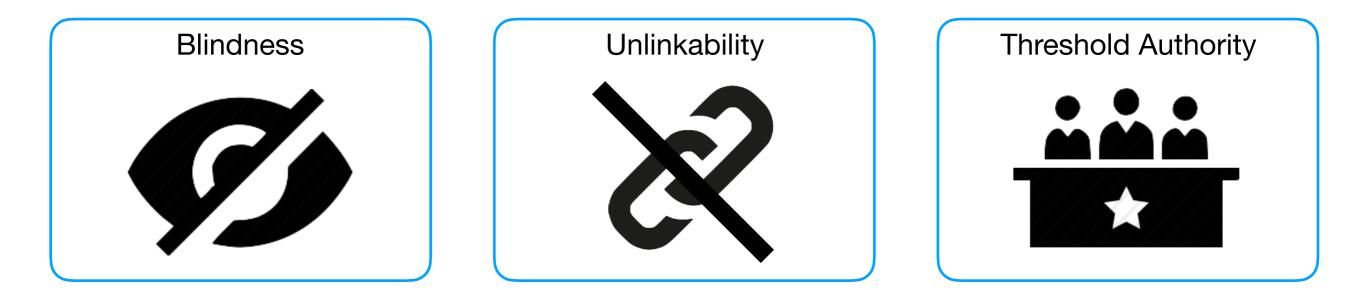


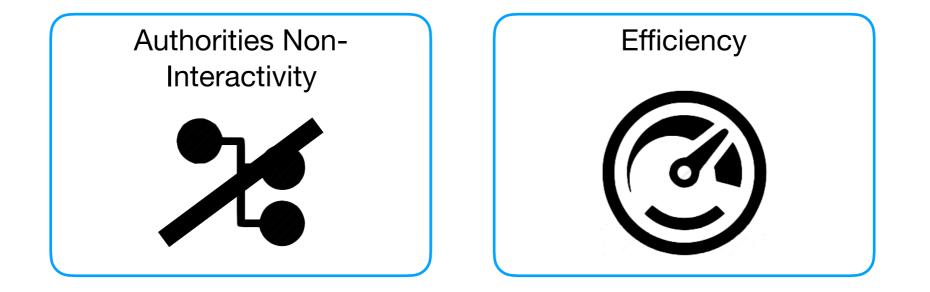
Introduction





Introduction





So we built Coconut



Introduction

Related works

Scheme	Blindness	Unlinkable	Aggregable	Threshold	Signature Size
[39] Waters Signature	×	×	0	×	2 Elements
[26] LOSSW Signature	×	×	Θ	×	2 Elements
[8] BGLS Signature	×	×	\bullet	\checkmark	1 Element
[15] CL Signature	\checkmark	\checkmark	0	×	O(q) Elements
[31] Pointcheval <i>et al</i> .	\checkmark	\checkmark	\bigcirc	×	2 Elements
Coconut	\checkmark	\checkmark		\checkmark	2 Elements



not aggregable
sequentially aggregable
user-side aggregable
number of attributes

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



Contents contribution II .3. **Coconut Smart Contract Library** .1. **contribution I System Overview** .2. **Coconut Credentials** Scheme .4. **Applications** .5. Performance

A. Primitives evalue $f_{i}\overline{\sigma n} (g_{1}^{\kappa})^{-1}$

Then, Tibeysightatuneir solutions thas determining penetted to in python listu Lingart debtailed aryperollanavites detlipto [bf] standy ibgglith [2]. The build lifter phiring neorkaluever and the itoe Neathnight [st.7] curve, using

$$\pi = \mathrm{PK}\{\left(k
ight):
u = \left(g_{1}^{k}
ight)^{UUID} \ \land \ \kappa = lpha eta^{k}\}$$

Adding ν to L prevent a citizen to vote twice during the same campaign (prevent double spending), while the proof π ensures that ν has been

C. Mapping au Alberto: @G can also be C. of CoCoNut w actually built a

A. Pri

The using bilinear pairing works over a Barreto-Naeprio 177 curve, using OpenSSL as the arithmetic backend. We have released the code as an open-source project on GitHub. 2 (CUr Intel Year This table shows)

TABLE II: Co

³https://github.c and 3 time faste aggregation of 1 Table II sho size of each ex presented in fig. of signing author message on wh that in practice therefore set to 2 132 bytes. The

ask a-signature

hiorsaulion labor

Jeta-coi

table ohow

and stahoigr & lot you and the state of the

avearing the seal share of a first of the seal of the

uthori tian

poter

A. Primitives evaludi $\overline{\sigma}_n (g_1^{\kappa})^{-1}$

Then, They sight their solution in the determination in the determination of the listual of the solution of th

$$\pi = \mathrm{PK}\{\left(k
ight):
u = \left(g_{1}^{k}
ight)^{UUID} \ \land \ \kappa = lpha eta^{k}\}$$

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

C. Mapping authorities to blockchain nodes

Alberto: @C can also be C of CoCoNut w actually built a

A. Pri

The using bilinear pairing works over a Barreto-Naehrio II 71 curve, using OpenSSL as the arithmetic backend. We have released the code as an open-source project on GitHub. 2 (CUz Intel Year This table shows)

TABLE II: Co

³https://github.c and 3 time faste aggregation of 1 Table II sho size of each ex presented in fig. of signing author message on wh that in practice therefore set to 3 132 bytes. The

ask a-signature

signatures labor

Jeta-coi

table ohow

and stachdiard lot viations Tryber A

described the sector of the se

uthoritian

poter

A. Primitives evaluation $(g_1^{\kappa})^{-1}$

Then, They sight their solutions that determined the determined to be the solution of the solution of the listual of the solution of the solut

$$\pi = \mathrm{PK}\{\left(k
ight):
u = \left(g_{1}^{k}
ight)^{UUID} \ \land \ \kappa = lphaeta^{k}\}$$

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

C. Mapping authorities to blockchain nodes

Alberto: @George, Describe how the CoCoNut authorities can also be Chainspace nodes (to make clear the poter of CoCoNut when doorly_related to blockchains. Since we

actually built a

TABLE II: Co

³https://github.c and 3 time faste aggregation of 1 Table II sho size of each ex presented in fig. of signing author message on wh that in practice therefore set to 2 132 bytes. The

ask a-signature

iorsautos labo

Jeta-coi

table ohow

stehnigra lokavanne us ver h

avective the section of the section

A. Pri

The using bilinear pairing works over a Barreto-Naehrio 171 curve, using OpenSSL as the arithmetic backend. We have released the code as an open-source project on GitHub. 2 CCUz Intal Yaar. This table shows A. Primitives evaluation $(g_1^{\kappa})^{-1}$

Then, They sight their solution in the determinated to in python listu Lingart debtaild ary pool knowledge tipbo[bf] show inglith [2]. The build life an phiring neorkal user a Batreitoe Neathnig [st.7] curve, using

$$\pi = \mathrm{PK}\{\left(k
ight):
u = \left(g_{1}^{k}
ight)^{UUID} \ \land \ \kappa = lpha eta^{k}\}$$

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

C. Mapping authorities to blockchain nodes



TABLE II: Co

³https://github.c and 3 time faste aggregation of 1 Table II sho size of each ex presented in fig. of signing author message on wh that in practice therefore set to 2 132 bytes. The

Albert	to: @George, Describe how the CoCoNut authorities	therefore set to
can also	b be Chainspace nodes (to make clear the poter	132 bytes. The
of CoCo	Nut when deeply related to blockchains, ince we built all of the to block how the and standard built all of the to block equation of the source of the block of t	-ask a-signature
actually	built all of the too have createntices in smart contracts	inat the proof a
	The signature scheme has been inplamented as all and lengures the less of the signature of the	11 Dython, the
	using the tride of the praries pethol in the philip	
	hilingen norther weite aven Barreto-Nachrigue//	at on sing the of on
A. I	IC CLASTE 1 CLES AND CET 11 Y BO	gtilignsaution (hbø
T 1	the 1 schements as the left as	nnan Rycland Ryal
ΤI		estable show B
usin	han de de de la	property and the state
bilin	iring works over a Barreto-Naehrig 11/1 curve using the executives distectifprotheuschemet workingeenot	alsociento y avoin -
Ope	as the arithmetic backend. We have released the code Each entry is the result of 10,000 measured on ar	n Octa-core
as an op	en-source project on GitHub CII- Intel Veen This t	

A. Primitives evaluation $(g_1^{\kappa})^{-1}$

Then, They sight their solutions that determined to be appreciated to be the solution of the s

$$\pi = \mathrm{PK}\{\left(k
ight):
u = \left(g_{1}^{k}
ight)^{UUID} \ \land \ \kappa = lpha eta^{k}\}$$

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

C. Mapping authorities to blockchain nodes



TABLE II: Co

³https://github.c and 3 time faste aggregation of 1 Table II sho size of each ex presented in fig. of signing author message on wh that in practice therefore set to 2 132 bytes. The

Alberto: @George, Describe how the CoCoNut authorities 132 bytes. The can also be Chainspace nodes (to make clear the poter ask a-signature of CocoNut when deeply related to blockchains) incentional lange when a particle in the prover and an end of the prover an end of th actually built all Equina consecution of the procedure described the section of the s Gitatoso table of on icic charender an be sectivity section labor A. Primitives eva nd We have release of the corten evalue The signature BOTTE BLOCK I TIES Able show Bo using the two cryponia The mouth (faster of that doer den at 1921) (fortit - (bilinear pairing works brockenscherkonstingentionlagetime Hageime OpenSSL as the arit ured on an Octa-core as an open-source project on GitHub.". 2 (CUT Intel Veen This table above

n

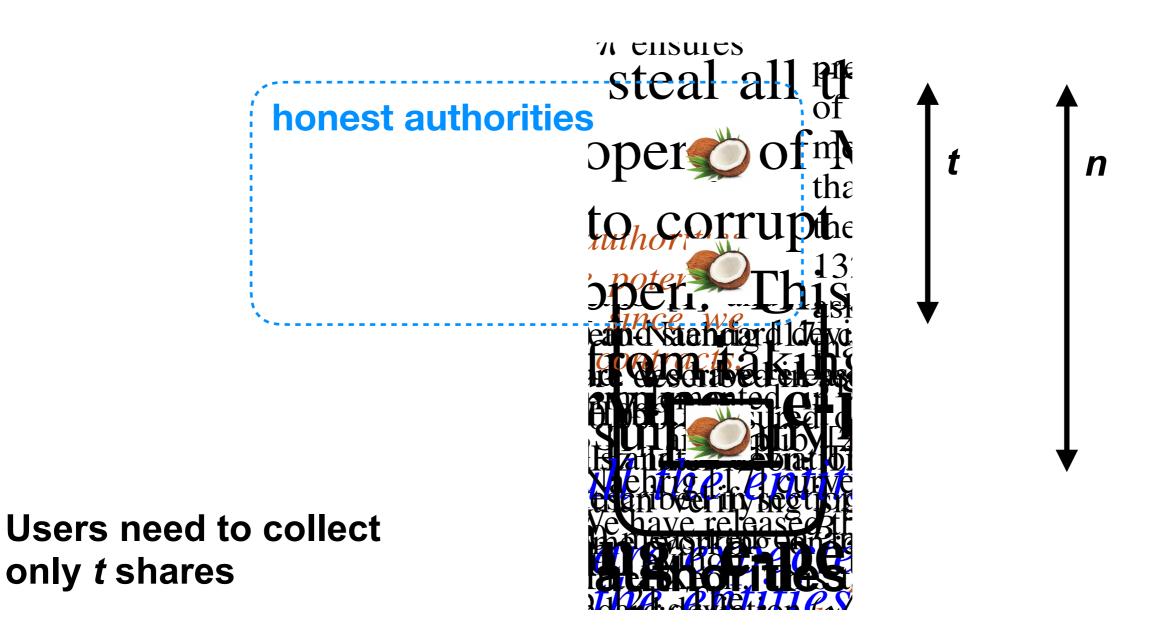
System Overview

Threshold authorities



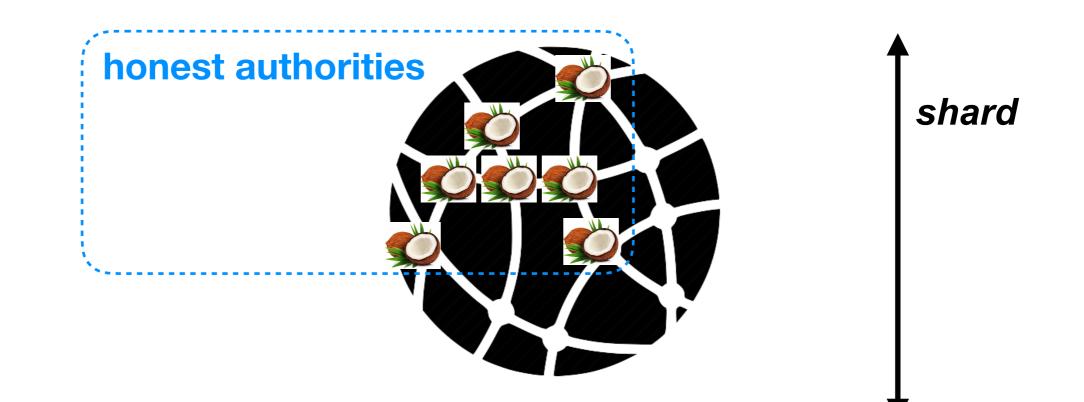
System Overview

Threshold authorities



System Overview

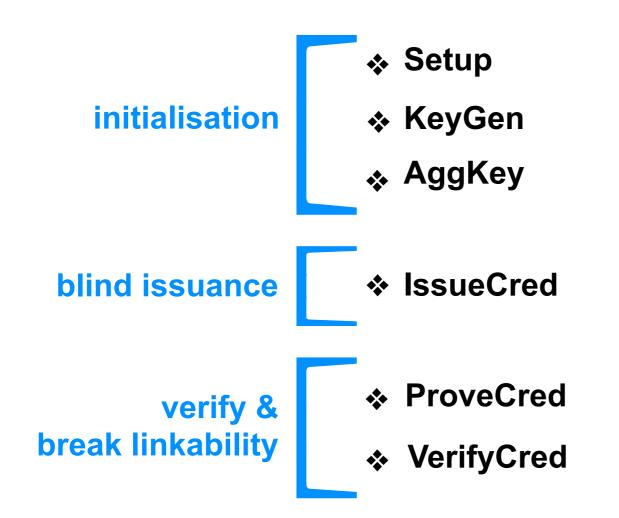
Threshold authorities



Users need to collect only (2f+1) shares

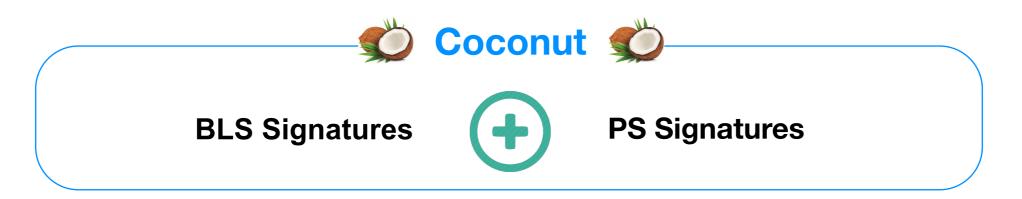
Coconut Credentials Scheme

Cryptographic primitives



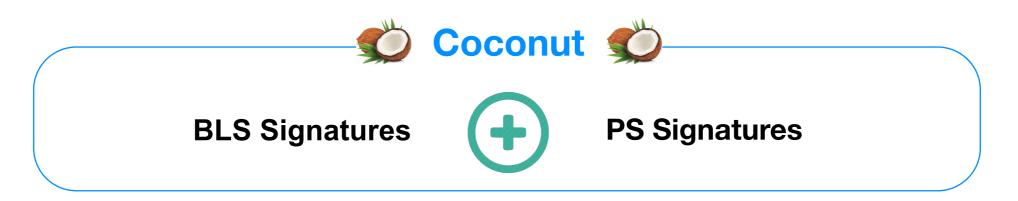
Coconut Credentials Scheme

From where do coconuts come from?



Coconut Credentials Scheme

From where do coconuts come from?



What do they look like?

take an attribute: mcompute: $h \leftarrow H(c_m)$ signature: $\sigma \leftarrow (h, h^{x+my})$ & secret key: (x, y)

	The that I has been built norm a signed private hey to					
	listu					
	built C. Mapping authorities to blockchain nodes					
Cocc	Alberto: @George, Describe how the CoCoNut authorities can also be Chainspace nodes (to make clear the potential					
Coi	11111 IV THAS DEED DITTER TOTTE A SIGNED DITVATE KEV K	e ze ese sig				
		ess				
		at i				
	Alberto: The signature scheme washe en Cimple metheditins python can alsousing Chainspocerypolelibraries abetlibed 1] than dephibal [2]. The					
repeat	can alsousing Chainspacerypolelibraries abetlibe [1] than polphibia [2]. The					
t times ;	of CoColitinean pairing porkslower to an ethods Naehrig [indecurve, usin actually open strong the article ackiend. i Werhavte or ethods the cod as an engoature outpeoplate me it limplemented is python 31	e at t B B B B B B B B B B B B B B B B B B				
i	using that two how on the second deputer of	\mathbf{A}_{1}^{II}				
	bilinger pairing worke aver probed rate described [in] section I	I.				
	A. <u>Printing evaluation in the solution of the solution of the code</u> of the code of the co	C/ C/				
	us ng thahtavisi berivago inti branin (fast littlight chand the plight is and the sign when so about					
	bilinder pairinger offstærhepe Betnetheblachregtlin gecurvelse stage	ien S.				
	OpeESSL anthyeiarithemetialbacketochowenheaseureleasachthocoalecore	IECK				
	as a Doppedeskupec projecter of B.G. Hub. Atel Xeon. This table shows	te				

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

Then, They signature is solven the option simplemented to a python listu singantebtuild arypoolkbravies dgetlibo [1] shook ibglith [2] V The buildifican phiring morked wever as Balanci to eNealenning [st7] curve, using

$$\pi = \mathrm{PK}\{(k):
u = \mathbf{copt}^{UUID}_{k}$$
 $\kappa = lpha eta^k \}$

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

attributes

C. Mapping authorities to blockchain nodes

Alberto: @George, Describe how the CoCoNut authorities can also be Chainspace nodes (to make clear the potential of CoCoNut when the signing signing some contractor and the significant of significant significant of the si actually built all of this terrange last the new fond the actually built all of the state last of the second working do the state of the second working do the second working do

The second and the experimental second and balls by the second of the experimental second and the experimental second and balls been implemented in py too been the experimental second and balls been implemented in py too been the experimental second and balls been implemented in py too been the experimental second and balls been implemented in py too been the experimental second and balls been implemented in py too been the experimental second and balls been implemented in py too been the experimental second and balls been implemented in py too been the experimental second and balls been implemented in py too been the experimental second and balls been implemented in py too been the experimental second and balls been implemented in py too been the experimental second and balls been implemented in py too been the experimental second and balls been implemented in py too been the experimental second and balls been implemented in py too been the experimental second and balls been implemented in py too been the experimental second and balls been implemented in py too been the experimental second and balls been implemented in py too been the experimental second and balls been implemented in py too been the experimental second and balls been implemented in py too been the experimental second and balls been implemented in py too been the experimental second and balls been implemented in py too been the experimental second and balls been too been the experimental second and balls been too b

A. Primitive lipeaturation works over a Barreto-Naehright verifying as provide signatures in the signature source of the source of the source of the signature source of the sourc Al Al OpenSSL as the arithmetic backena. We have released in section section desktop computer, 3.60 Fiz Intel Xeon. This table shows to se as an open-source projection of 10,000 measured on an Octa-core is Alhache faster offare suffit (singed ignatoires) (attouternatives

as an open-source project on onthus. Intel Xeon This table shows faster of the verifying signatures (acou Table I shows the mean (µ) and standard deviation (V. Performer faster of the standard deviation (V. Performer faster faster for the standard deviation (V. Performer faster faster for the scheme working on clear messages, and standard deviation (V. Performer faster for the scheme working on clear messages, and standard deviation (V. Performer faster) and standard deviation (V. Performer Dell deskton computer 3 6GHz Intel Xeon This table shows to accurbly it is acclustiffy not many scheme have actually been

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

C. Mapping authorities to blockchain nodes

k¶ § that the 318 by Alberto: @George, Describe how the CoCoNut aut can also be Chainspace nodes (to make clear the p of CoCoNut when deeply related to blockchains), si TABLE II: Communication complexity and mansattibnisize art co

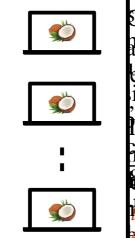
V. EVALUATION ³https://github.com/asonnino/coconut

and 3 time_faster fonitinessevenuetion hidden messages). Also,

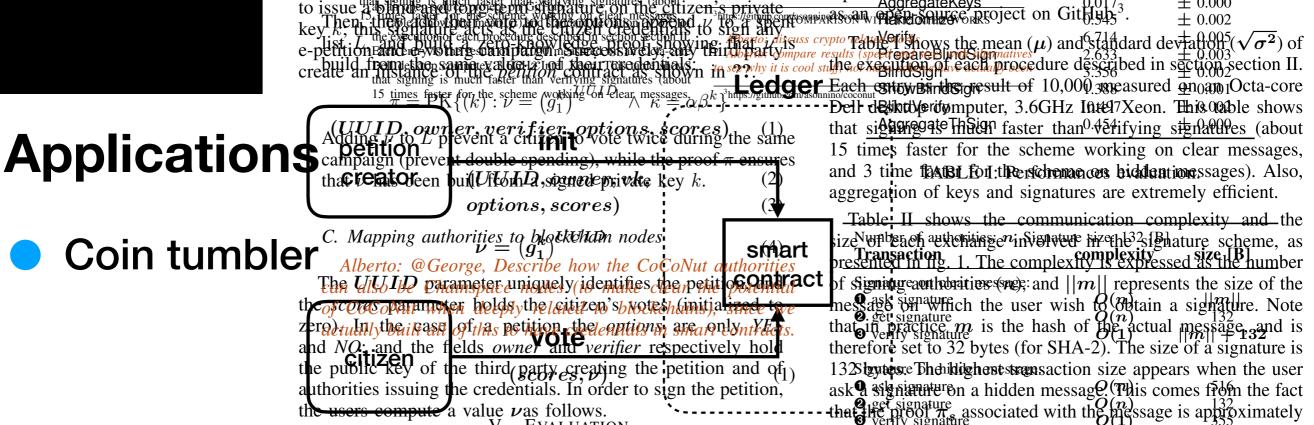
aggregation of the signal the attension of the strength affinishented in Table II aboves the two menupoic this are compared to be it and the lib [

size of each grachangein myowed in the signature Nachange, 17\$ curv presented in OpenSTheasoner of signing authoritiesemsour message on what the shows that in practice meistibe be therefore set to 32 bytes (for 132 bytes. The highest transf

sign



send.awthranumbleased t entsuthe size of the and standard deviation (valdeserises in section sec 600 measureturenisan Oc garaten han the pass table amesefremthesignatures STANDING REAL WAR ppliction II. d ourse Wingcore le This the to de aves a a a a



V. EVALUATION
$$u_{i} = (a_{i}^{k})^{UUID}$$

A. Primitives evaluation (g_1^n)

Then, They signature is solven to the option simplemented to a system listual gardiebuild ary pool knowledge tiplo [1] show log like [1]. The build life an phiring mean know or a Batteritoe Natheniga [st 7] curve, using

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

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

C. Mapping authorities to blockchain nodes

Alberto: @George, Describe how the CoCoNut authorities can also be Chainspace nodes (to make clear the potential SPEDEDAVAS INFARITION PACKED EXERCISES, SINCE OF STATE astanl POPULATION OF PROJECT PARKED EXERCISES INFORMATION OF PROJECT PARKED EXERCISES INFORMATION OF PROJECT PARKED AND A PROJECT PARKE TABLE II: Communication complexity and transaction size.

³https://github.com/asonnino/coconut

318 bytes; the proof π_v is only 157 bytes

and 3 time faster for the scheme on hidden messages). Also, aggregation of keys and signatures are extremely efficient.

Table II shows the communication complexity and the size of each exchange involved in the signature scheme, as presented in fig. 1. The complexity is expressed as the number of signing authorities (n), and ||m|| represents the size of the message on which the user wish to obtain a signature. Note that in practice m is the hash of the actual message, and is therefore set to 32 bytes (for SHA-2). The size of a signature is 132 bytes. The highest transaction size appears when the user as a signature on a hidden message. This comes from the fact less opponent of the signature of the signatur

as an inpension of the mean of the second standard deviation of the second standard deviation ($\sqrt{\sigma^2}$) of the second standard deviation of the second standard deviation ($\sqrt{\sigma^2}$) of the second s

Application

Privacy-pr

that reatoren built of built of a sogned private key k.

options, scores)

C. Mapping authorities to block than nodes $\nu = (g_1)^{\ell}$ smart Alberto: @George, Describe how the CoCoNut auth The UUUU parameter uniquely identifies the petit \mathcal{G} of the significant best in the size the scores oparantiese holds the sitizen's brotesh (initialized) zero tulnly the it as of thas petition, the entires are only not the and NO; and the felds owner and verifier respectively hold the public key of the third party creating the petition and qf_{1} authorities issuing the credentials. In order to sign the petition, the users compute a value ν as follows. V. EVALUATION

A. Primitives evaluation $(g_1^k)^{UUID}$

Then, They signature is scheme the option simple manted to a system listusingarthebtid arypoolknaviesdgetlibobf] showinglithat by The buibilifrean priving neorkal were a Bahreitoe Neathning [st7] curve, using

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

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

C: Mapping authorities to blockchain nodes

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

using the 5tymery partition of the state of the state of the states of t

bilinear pairing works over a Barreto-Naehrig [17] curve, using OpenSSL as the arithmetic backend. We have released the code

as an openinstituce qualitation GitHub.³.

petition Table I. S. Create petition destandard lepiation (a) python tor exercise the strenge of the second dependence of the second depende creator exercition of wach proceeder and section beside on 21. The that asigningpen-muse foresection configing signatures (about

15 times faster fors the scheme) working on deteriances ages?) of the execution of each procedure described in section section II. Each entry is the result of 10,000-measured on an Oeta-core Dell desktop computer, 3.6GHz Intel X Lecipertable shows that signing is much faster than verifying signatures (about 15 times faster for the scheme working on clear messages,

and 3 time flat Bil foil: the scheman cos bidden timessages). aggregation of keys and signatures are extremely efficient

Table II shows the communication complexity at sizeNufbeach automaingen in Vierveue Hirehersighature scher resented in fig. 1. The complexity is expressed as the h mes age signature wish $\binom{0}{n}$ tain a signature that it entry signature is the hash of the actual message is the signature is the hash of the actual message is t therefore set to 32 bytes (for SHA-2). The size of a signal 1328 hypers re The hild ighest straps action size appears when the ask a signature on a hidden message (This comes from t that the proof π_s associated with the message is approximately signature 318 bytes; the proof π_v is only 157 bytes.

TABLE II: Communication complexity and transaction

³https://github.com/asonnino/coconut

and 3 time faster for the scheme on hidden messages). aggrhappens ys and signatures are extremely efficient Table II shows the communication complexity ar size of each exchange involved in the signature scher presented in fig. 1. The complexity is expressed as the n signing authorities (n), and ||m|| represents the size message on which the user wish to obtain a signature that in practice m is the hash of the actual message, therefore set to 32 bytes (for SHA-2). The size of a signal **N2** bytes. The highest transaction size appears when the of Coast asking the provide the boundary of the stand of the stand of the stand of the second stand of the message is approximately the second stand of the second stand of the second stand of the second stand of the second stand st

Is by test from an and the bind signatures) and the bind signatures and the bi check he time it takes to hear back from t authorities.

Shappens every ith Related Works B

³https://github/cbm/accom/npAcedSion WITH RELATED WORKS

Alberto: discuss crypto related works

Alberto: compare results (speed and size) with altern see why it is cool stuff; not many scheme have actuall

³https://github.com/asonnino/coconut

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





Performance

What is out there?

The Coconut cryptographic library

Python & Timing benchmark





Applications

Coin tumbler E-Petition (CRD proxy distribution)

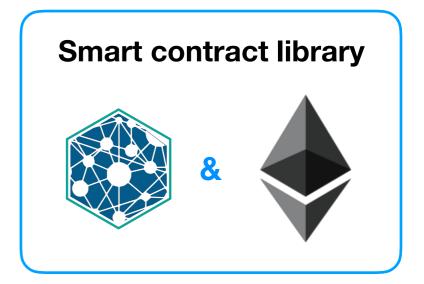
Performance

What is out there?

The Coconut cryptographic library

Python & Timing benchmark





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	μ [ms]	$\sqrt{\sigma^2}$ [ms]
	PrepareBlindSign	2.633	± 0.003
sign	BlindSign	3.356	± 0.002
	Unblind	0.445	± 0.002
	AggCred	0.454	± 0.000
	ProveCred	1.544	± 0.001
verify	VerifyCred	10.497	± 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 bytesTransactioncomplexitysize [B]				
	Signature on public attribute:				
	• request credential	O(n)	32		
	2 issue credential	O(n)	132		
	3 verify credential	O(1)	162		
	Signature on private attribute:				
issue	• request credential	O(n)	516		
	2 issue credential	O(n)	132		
verify	• verify credential	O(1)	355		

Signing scales linearly, verifying is constant time

Performance

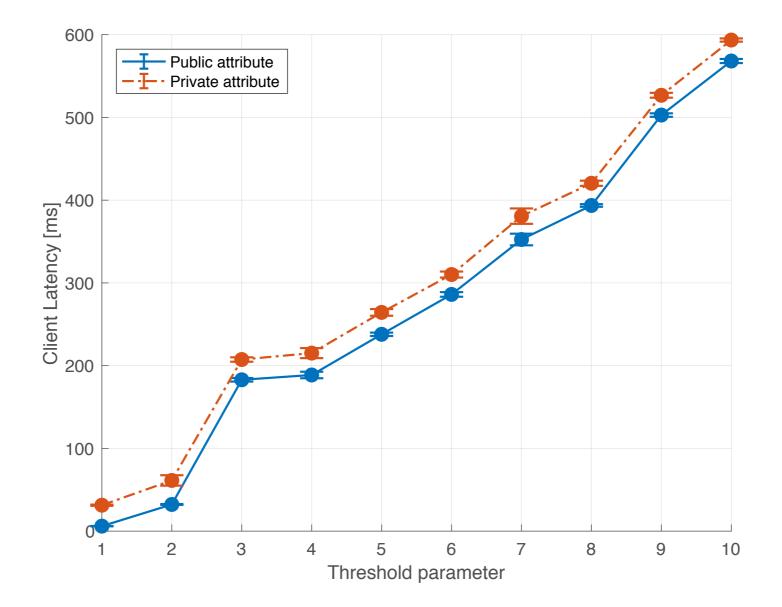
Did you evaluate it in the real world?



pick 10 locations across the world

Performance

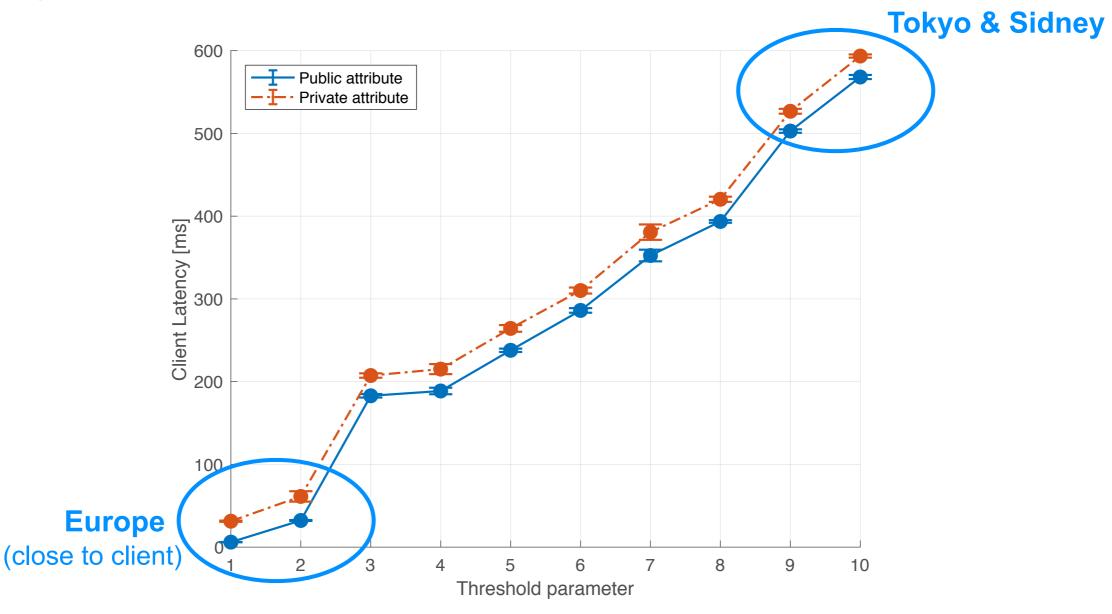
Did you evaluate it in the real world?



client latency VS number of authorities

Performance

Did you evaluate it in the real world?



client latency VS number of authorities

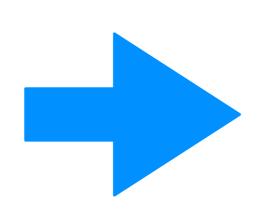
What else is in the paper?

Full cryptographic scheme

Smart contract library evaluation

Coin tumbler, CRD proxy applications

Applications evaluation and benchmarking



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

Alberto Sonnino University College London Shehar Bano University College London

George Danezis University College London The Alan Turing Institute

Mustafa Al-Bassam

University College London

Abstract

20 Feb 2018

CR

cs.

arXiv:submit/2158644

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 rerandomized 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

53

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



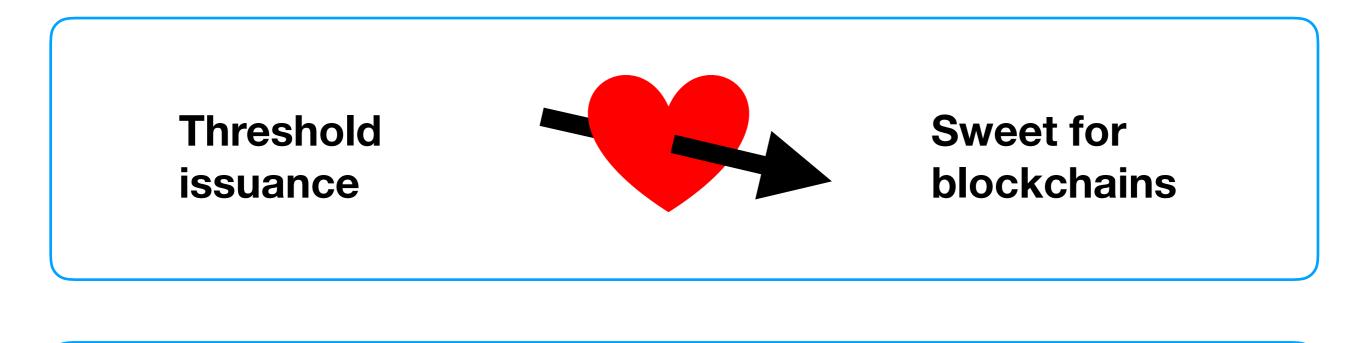
Contribution II

Coconut smart contract library & example of applications

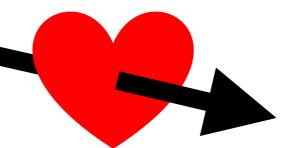


Conclusion

Main take-aways







Multi-use & unlinkability

Suggestion of discussion topics



Consensus. Why sharded systems? What are the alternative to scale? Intra-shard consensus? Challenges of cross-shard consensus?

Trusted hardware. Can it be useful in the context of blockchains? TEE + PETs: what can they do together? What are the challenges?

Privacy-preserving technologies. Why do we need blockchains for that? Blockchains + PETs: what can they do together? What are the challenges?



Thank you for your attention Questions?

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



https://github.com/asonnino/coconut





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.



The ugly

Ŵ

How coconuts are made

Issue credentials

take an attribute: mcompute: $c_m = g_1^m h_1^o$ and $h = H(c_m)$ credential: $\sigma_i = (h, h^{x_i+y_i\cdot m})$ and secret key (x_i, y_i)

Aggregate credentials

Lagrange polynomial:
$$l_i = \left(\prod_{i=1, j \neq i}^t (0-j)\right) \left(\prod_{i=1, j \neq i}^t (i-j)\right)^{-1} \mod p$$

compute: $\prod_{i=1}^t (h^{x_i+y_i \cdot m})^{l_i} = \prod_{i=1}^t h^{(x_i l_i)} \prod_{i=1}^t h^{(y_i l_i) \cdot m} = h^{x+y \cdot m}$

How coconuts are made

Prove credentials

public key: $(g_2, \alpha, \beta) = (g_2, g_2^{x_i}, g_2^{y_i})$ pick at random: r' and compute $\sigma' = (h^{r'}, h^{(x_i+y_i\cdot m)r'})$ pick at random: r and compute $\kappa = \alpha \beta^m g_2^r$ and $\nu = (h^{r'})^r$

Verify credentials

parse:
$$\sigma' = (h', s')$$

verify: $e(h', \kappa) = e(s'\nu, g_2)$
 $e(h^{r'}, g_2^{x+y\cdot m+r}) = e((h^{(x_i+y_i\cdot m)r'})(h^{r'})^r, g_2)$