Ouroboros Crypsinous
Privacy-Preserving Proof-of-Stake

Thomas Kerber
t.kerber@ed.ac.uk

Aggelos Kiayias
akiayias@ed.ac.uk

Markulf Kohlweiss
mkohlwei@ed.ac.uk

Vassilis Zikas
vzikas@ed.ac.uk

The University of Edinburgh & IOHK

May 17, 2019

Ouroboros Crypsinous
Privacy-Preserving Proof-of-Stake

Thomas Kerber
t.kerber@ed.ac.uk

Aggelos Kiayias
akiayias@ed.ac.uk

Markulf Kohlweiss
mkohlwei@ed.ac.uk

Vassilis Zikas
vzikas@ed.ac.uk

The University of Edinburgh & IOHK

May 17, 2019

Introduction

- Distributed ledgers allow users to agree on sequences of blocks.
- Users can append blocks to the sequence under some conditions.
- In proof-of-stake, this depends on their stake – their money in the system.

Motivation

- Proof-of-stake has advantages over proof-of-work:
  - More environmentally friendly.
  - Less susceptible to external attacks.
- However, constructions rely on knowing the “stake” each party has.
- We construct a proof-of-stake system working with a Zerocash-like transaction system, based on Ouroboros Genesis.

Our Contributions

- We construct the first[1] formally proven privacy-preserving proof-of-stake protocol.
- We model and prove this privacy secure in the UC setting.
  - The full UC specification can be found in the paper.
- We preserve the important adaptive security guarantees of the parent protocols, by using different and novel forward-secure primitives.
  - We utilise a SNARK-friendly hash-based construction in place of forward-secure signatures.
  - We define and use key-private forward-secure encryption.

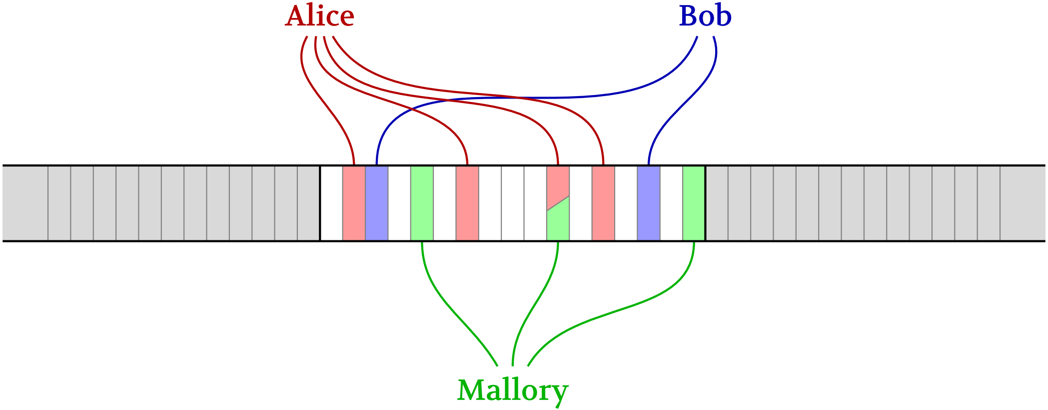
---

[1] There is concurrent and independent work by Ganesh et al. on the same subject.

Background – Ouroboros Genesis

- Time is divided into discrete units: large epochs, and small slots.
- When an epoch starts, its entropy η is determined.
- In every slot sl, stakeholders evaluate a VRF at (η, sl).
- If the result falls under a target, determined by their stake, they create a block.

Background – Ouroboros Genesis



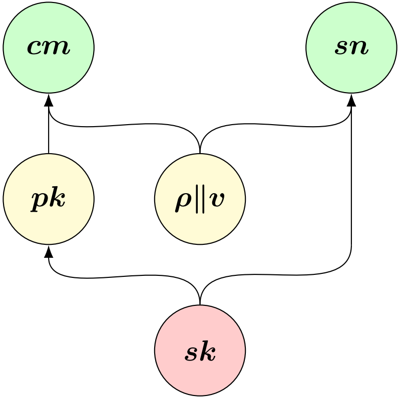
Background – Zerocash

- Bitcoin maintains a set of unspent coins.
- This leaks a lot about transactions.
- Transactions generally insert and delete some coins.
- Zerocash separates this, and maintains sets of created coins, and destroyed coins.

Background – Zerocash

- To make these unlinkable, the sets store different cryptographic properties of the same coin.
- To spend, you prove membership in the set of created coins, and non-membership in the set of destroyed coins.
    - Membership is proven by Merkle-tree membership proofs.
    - Non-membership is proven by revealing.
- This is done in zero-knowledge, along with proofs of consistency properties, such as transactions being zero-sum.

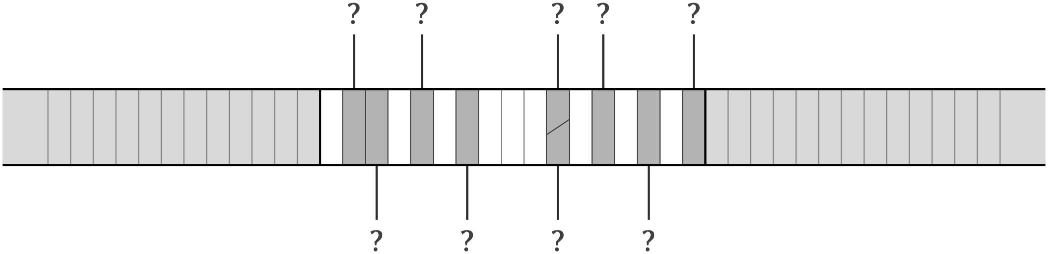
Background – Zerocash



Protocol – Crypsinous in a Nutshell

- We run variants of Ouroboros Genesis and Zerocash together.
- We move Ouroboros Genesis' leadership proof into zero-knowledge.
- We prove our stake with a one-to-one Zerocash transfer.
- The VRF is replaced with a zero-knowledge PRF evaluation.
- There are a number of subtle problems however...

Protocol – Crypsinous in a Nutshell



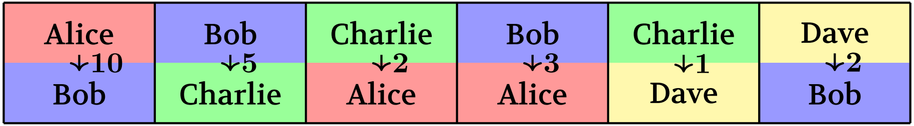
Protocol – “Frozen” Stake Distributions

- Ouroboros Genesis requires stake to be unchanging during an epoch, to prevent grinding attacks.
- By doing one-to-one transactions, we must change it.
- We also cannot prevent users from spending.
- We maintain sets of leadership-eligible and spending-eligible coins.
- Spending a coin removes it from leadership for the epoch.
- One-to-one leadership proofs create their new coin deterministically.

Model

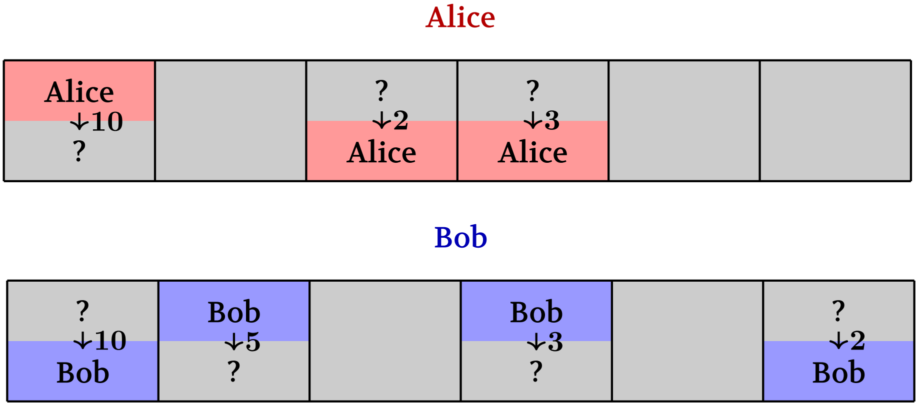
- Zerocash is not UC secure.
- Existing ledger functionalities are insufficient for privacy-preserving transactions.
- We introduce a private ledger $\mathcal{G}_{\mathsf{PL}}$, and parameterise it to implement privacy-preserving transactions.

Model – Public Ledger



Alice	Bob	Charlie	Bob	Charlie	Dave
↓10	↓5	↓2	↓3	↓1	↓2
Bob	Charlie	Alice	Alice	Dave	Bob

Model – Private Ledger



Alice

Alice ↓10 ?		? ↓2 Alice	? ↓3 Alice		

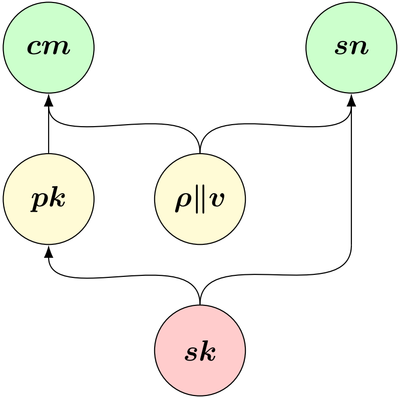
Bob

? ↓10 Bob	Bob ↓5 ?		Bob ↓3 ?		? ↓2 Bob

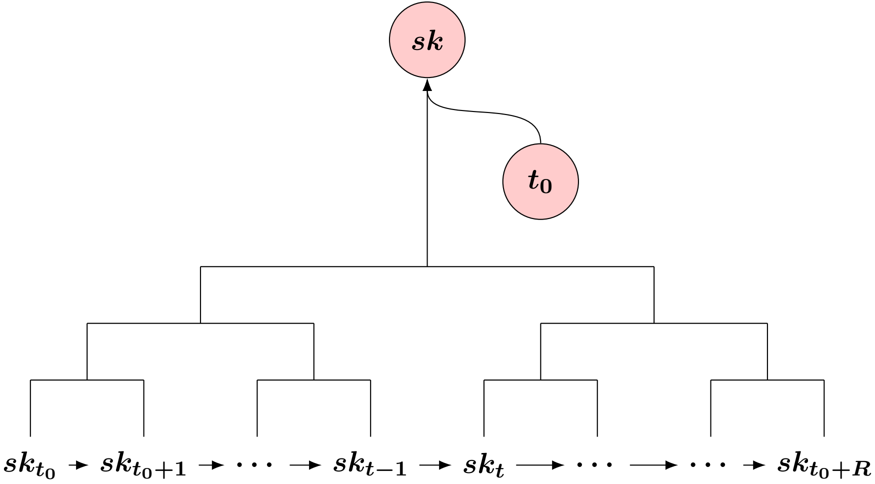
Adaptive Security – Leadership Proofs

- For adaptive security, honest slots should not later fall into adversarial control.
- Ouroboros Genesis uses forward-secure signatures, which are too heavy for being used within zero-knowledge.
- We use a combination of Merkle-tree membership proofs and key erasure to construct a lightweight replacement.

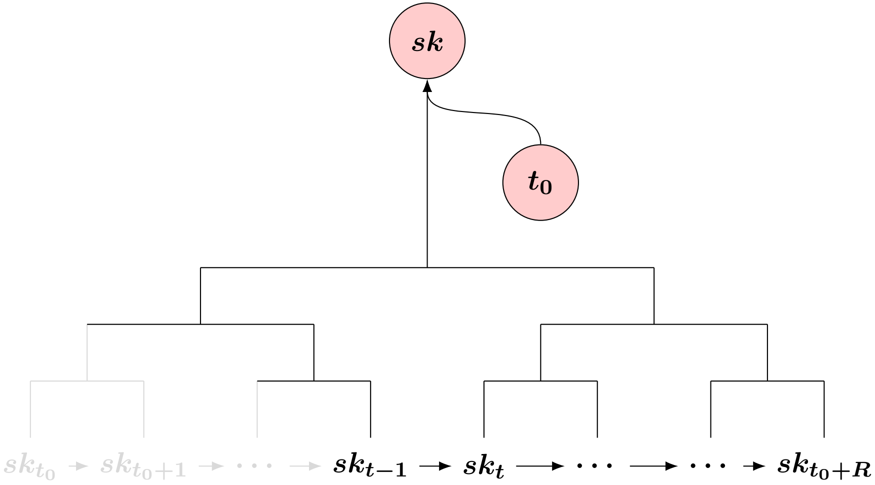
Adaptive Security – Recall: Zerocash



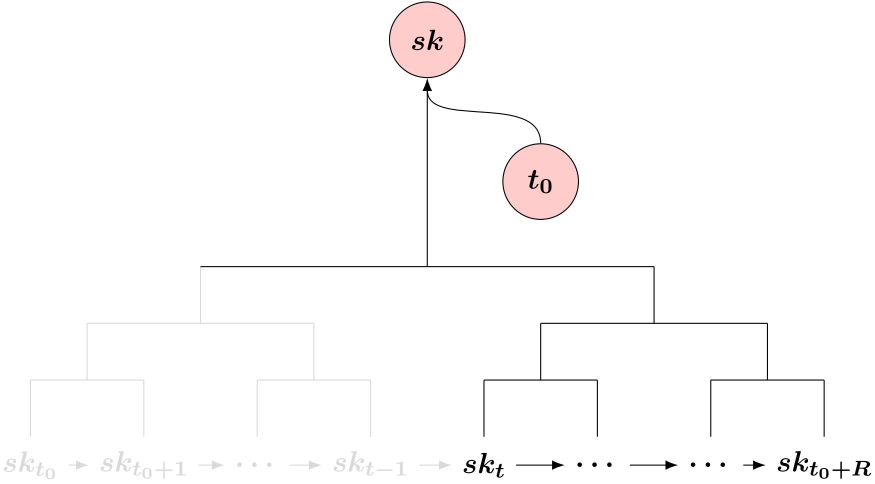
Adaptive Security – Leadership Proofs



Adaptive Security – Leadership Proofs



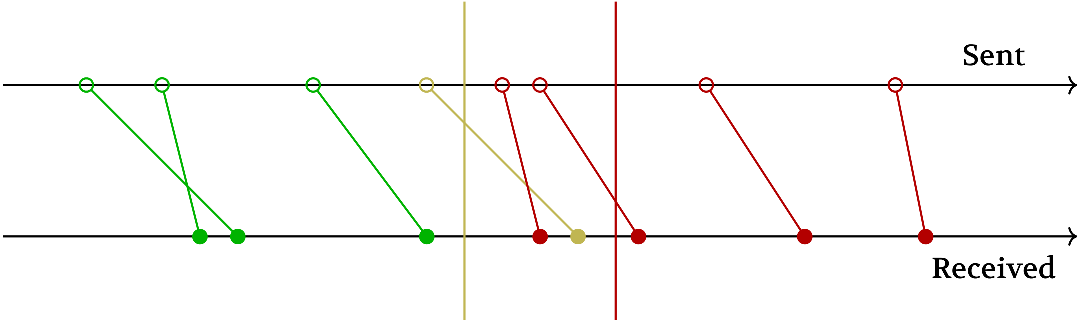
Adaptive Security – Leadership Proofs



Adaptive Security – Non-Committing Encryption

- Zerocash requires (key-private) encryption.
- Adaptive corruption requires encryption to be non-committing.
- Non-committing encryption is expensive.
- We employ key-private forward-secure encryption.

Adaptive Security – Non-Committing Encryption



Conclusion

- We construct a privacy-preserving proof-of-stake protocol.
- We prove it secure in UC, with adaptive corruptions
- We model the private ledger, and use it to construct a private currency.

Performance – SNARK Gate Estimation

Constraint count	Transfers	Lead
Check $\textsf{pk}_{c_i}$	$2 \times 27{,}904$	$27{,}904$
Check ρ_{c_2}, $\textsf{sk}_{c_2}$		$2 \times 27{,}904$
Path for $\textsf{cm}_{c_i}$	$2 \times 43{,}808$	$43{,}808$
(1 layer of 32)	$(1{,}369)$	$(1{,}369)$
Path for $\textsf{root}_{\textsf{sk}_{c_i}}$		$34{,}225$
Check $\textsf{sn}_{c_i}$	$2 \times 27{,}904$	$27{,}904$
Check $\textsf{cm}_{c_i}$	$4 \times 2{,}542$	$2 \times 2{,}542$
Check $v_1 + v_2 = v_3 + v_4$	1	
Ensure that $v_1 + v_2 < 2^{64}$	65	
Check y, ρ		$2 \times 3{,}252$
Check (approx.) $y < \textsf{ord}(G)\phi_f(v)$		256
Total	$209{,}466$	$201{,}493$

Network Anonymity – The Problem

- We assume fully adversarial networks.
- The adversary can show different chains to different users.
- He can tell which chain is being extended.
- Therefore the leader is leaked.

Network Anonymity – Weaker Threat Models

- Mixnets solve this.
- The leadership anonymity of Crypsinous upgrades gracefully.
- Mixnets are not practical in this setting.
- More practical models, such as TOR, are challenging to model, and not our focus.