Transaction privacy: past, present, and future

World Crypto Con, Las Vegas

Sarang Noether, Ph.D.

Halloween 2019

Disclaimers

The views expressed in this presentation are solely those of the author, and do not necessarily reflect those of any other person, organization, or community.

The material in this presentation is for educational purposes only and should not be construed as financial, legal, or other professional advice, nor as endorsement of any kind.

The author is an independent research mathematician who contributes to the Monero Research Lab workgroup and receives funding support from the Monero community.

Goals

You should understand and appreciate:

- the **importance** of fungibility and privacy in transaction protocols
- the **evolution** of cryptographic approaches used
- **differences** between transaction models and proving systems
- how **tradeoffs** in efficiency and trust affect design choices

The math gets spooky; you'll find none of it here.

Fungibility?

Privacy and fungibility are related. **Fungibility** can mean indistinguishability. Can particular funds be blacklisted?

Transparent ledgers like Bitcoin are highly non-fungible and non-private.

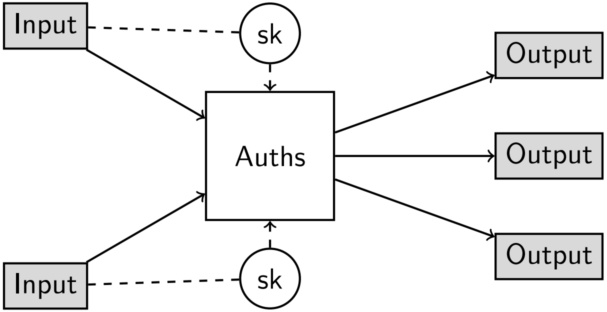
- Addresses appear on the chain as part of transactions
- Amounts are in the clear
- Assets have distinguishable history and properties

₿ $\neq$ ₿

Transaction fundamentals

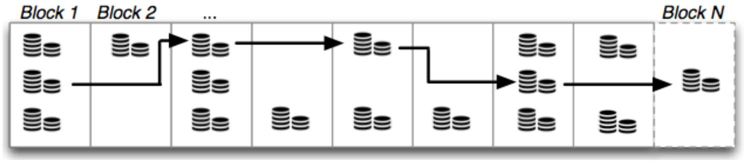
Transactions (primarily) consist of **inputs**, **outputs**, and **authorizations**.

A transaction consumes inputs using authorizations, and generates new outputs. A digital signature is the most basic authorization.



Transaction graph

Any network participant can analyze a transparent ledger and examine the flow of funds through transactions.



A lot of research has focused on methods of chain analysis using transparency.

Why care?

Transparency and non-fungibility have consequences that you may or may not like:

- **Censorship**: miners and other entities can censor transactions based on addresses, amount, location, or other transaction data/metadata
- **Markets**: certain types or quantities of assets may fetch a market premium based on history or usage
- **Linking**: any third party with access to the public chain can link data and amounts and attempt to link to identity, increasing personal risk
- **Competition**: business competitors can trivially gain information about transactions that yields proprietary information or business data
- **Regulations**: certain jurisdictions may impose rules requiring financial information be kept private in storage and transit

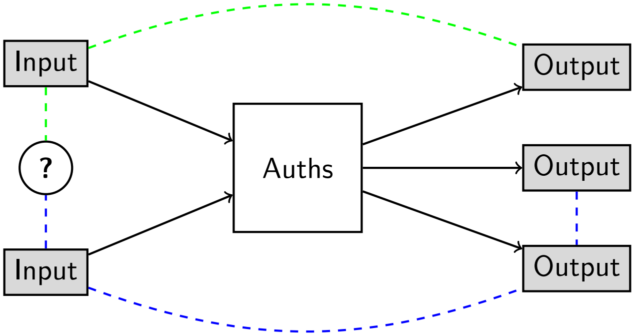
The term “privacy coin” has it reversed! Shouldn’t all bathrooms have doors?

Spectrum of common approaches

$$\begin{tikzcd}
\text{Transparent mixers} \arrow[d] & \\
\text{Specified ambiguity sets} \arrow[d] & \text{Mimblewimble} \\
\text{Accumulators} &
\end{tikzcd}$$

Example: transparent mixers

Transparent mixers take transparent funds with possibly different owners and include them in the same transaction. Authorizations can be modified accordingly, but typically no new math.



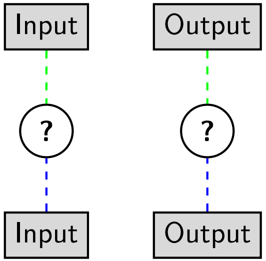
Example: transparent mixers

Upsides:

- Breaks assumption of common ownership

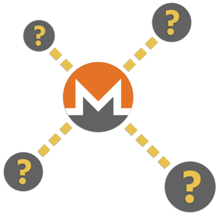
Downsides:

- Little-to-no privacy for senders or recipients
- Interactive, opt-in process
- Does not provide fungible assets, only uncertainty
- Many analysis methods still apply as usual



Example: RingCT

RingCT is a protocol that uses a set of constructions, implemented in Monero and elsewhere. It uses one-time addresses to mitigate (but not eliminate) linking. Ambiguous signatures sign on behalf of a sender-chosen group of possible spends. Amounts are hidden using Pedersen commitments.



Taken together, these provide limited sender ambiguity, recipient anonymity, and amount hiding toward fungibility.

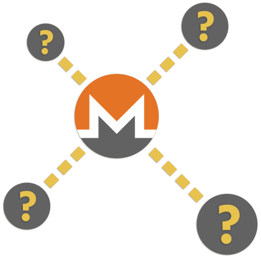
Example: RingCT

Upsides:

- Replaces undeniable signatures with ambiguous signatures
- Hides amounts
- Obscures transaction graph

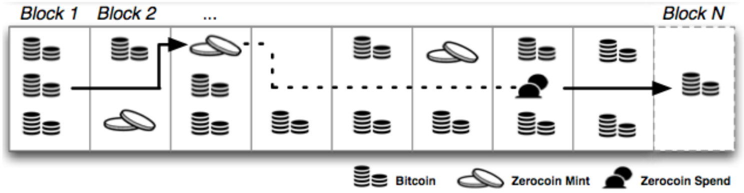
Downsides:

- Metadata can reduce effective anonymity
- Scaling is inefficient
- Does not fully remove common-ownership heuristic



Example: Zerocoin

Zerocoin is/was the first serious accumulator approach to fungibility as an extension to transparent ledgers. It was used in assets like Zcoin (not anymore!) but is considered broken.



You **mint** a Zerocoin by burning known Bitcoin and adding to a coin accumulator. You **spend** a Zerocoin by proving ownership of an unknown accumulator coin, transferring to Bitcoin.

Example: Zerocoin

Upsides:

- Accumulator ensured maximal spend ambiguity
- Could play nicely with transparent assets

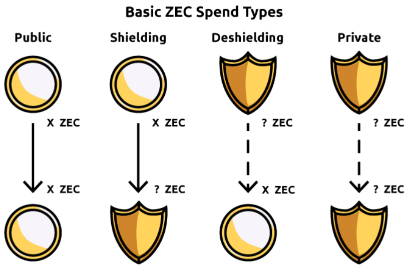
Downsides:

- Possible to burn honest users' funds upstream
- Flawed mint proof exploited for inflation
- Limited to fixed amounts with no direct transfer
- Large and inefficient proofs
- Timing and linking carry over to main chain

Zerocash/Zcash

Zerocash was a transaction protocol that influenced the **Zcash** protocols. Current protocols use Merkle tree accumulators, but with much greater flexibility than Zerocoin due to more robust proof systems.

Transactions support direct transfers of hidden amounts, hiding spends within the accumulator. Privacy is optional.



Example: Zerocash/Zcash

Upsides:

- Accumulator ensures maximal spend ambiguity
- Can play nicely with transparent assets
- Supports direct transfers
- Small proofs with verification efficiency

Downsides:

- Proving system soundness requires non-collusion and correct MPC
- Complex structure due to more general proving systems
- Optional privacy
- Transitions admit linking and timing analysis

Why so many approaches?

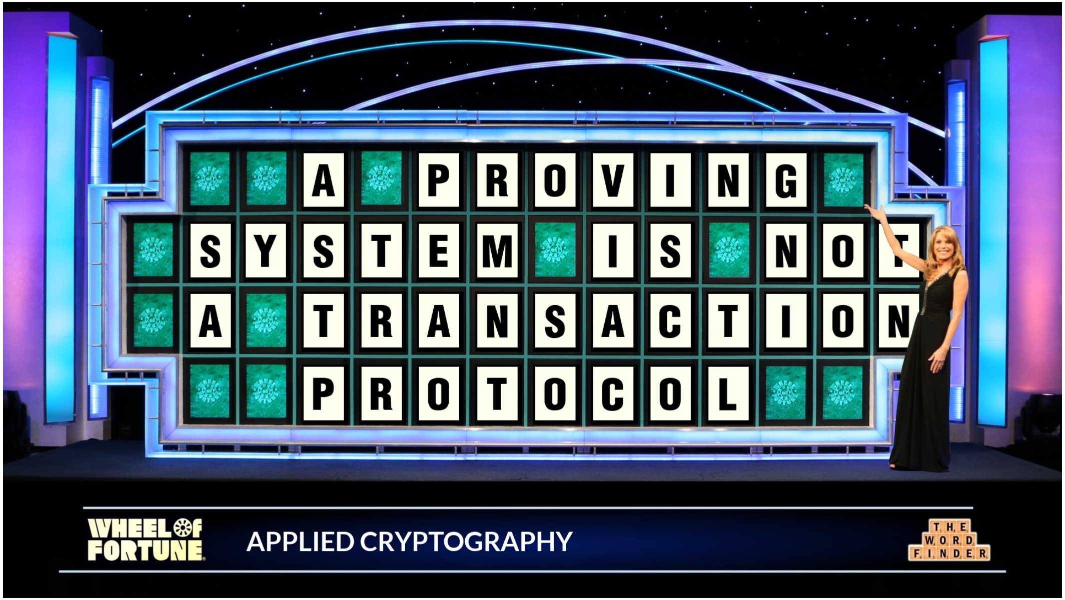
- **Transparent mixing** offers effectively *no* privacy and very limited fungibility, but retains general compatibility with transparent ledgers using interactive processes.
- **Obfuscation/ambiguity** designs can provide reasonable sound privacy and fungibility, but at the cost of efficiency and some graph analysis.
- **General proving systems** can be used to built small and fast transactions for accumulator-based transaction models with good privacy, but practically at the cost of soundness guarantees.

Transaction model vs. proving system

A **transaction model** is a collection of cryptographic constructions (signatures, proofs) used to demonstrate things like asset ownership, non-double-spending, balance, destination, and so on. We've seen several.

A **proving system** is a cryptographic construction used to prove and verify mathematically-defined statements involving private and public data, ideally in zero knowledge. Some work on specific statements, and others are more general.

- Having a proving system does not give you a transaction model. A language is useless if you have nothing to say.
- The constructions in some transaction models can be built using proving systems to produce and verify transactions, but this has many caveats relating to efficiency and specificity. The nature of the reference string plays a large role (e.g. for Zcash and related assets).


A PROVING
SYSTEM IS NOT
A TRANSACTION
PROTOCOL
WHEEL OF FORTUNE
APPLIED CRYPTOGRAPHY
THE WORD FINDER

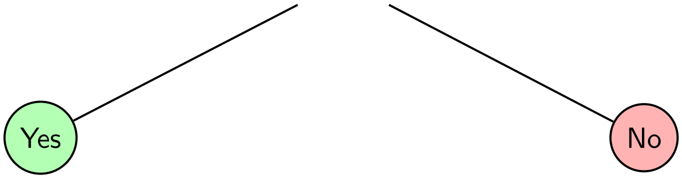
Metadata matters

- Network/location data
- Transaction relay
- Input/output structure
- Timing/periodicity
- Address linking
- Migrations/transitions
- Side-channel attacks

The best transaction models in the world do not remove all metadata!

Research directions

Are you willing to offload soundness to third parties?



There are good proving systems on which you can build fast and efficient transaction protocols for accumulator-based anonymity and privacy.

The options are mostly ad-hoc and suffer from scaling problems for signature and proof sizes. Verification typically scales as least as badly as size.

Example: MMORPG (Zcash Sapling)

MMORPG is a variation by Bowe *et al.* of a proving system by Groth. It proves statements about circuit satisfiability in zero knowledge.

A structured/trusted setup is required, but is distributed. Proof size is less than 200 bytes for any circuit complexity. Proving is linear in the circuit complexity, but verification is linear only in the witness complexity.

For the Zcash Sapling circuit, proving time is $O(1)$ seconds, and verification time is $O(1)$ milliseconds. (Proving time here relies on non-general circuit optimizations!)

Used in:

Example: Bulletproofs

Bulletproofs are an extention by Bünz *et al.* of an underlying proving system by Bootle *et al.* to prove statements about circuit satisfiability in zero knowledge.

There is no structured or trusted setup; reference strings are generated publicly. Size is logarithmic (sublinear) in the circuit complexity, but proving and verification are linear.

For estimates of the Zcash Sapling circuit, proving time is $O(10)$ seconds and verification time is $O(1)$ seconds (down to $O(100)$ milliseconds) for a proof of size $O(1)$ kB.

Used in (optimized range proving only):

Wild oversimplification

	Bowe[1]	Bünz[2]	Ben-Sasson[3]
Trust-free	✗	✓	✓
Small proof	✓	✓	✗
Fast proving*	✗	✗	✗
Fast verifying	✓	✗	✓

* highly dependent on implementation and circuit

[1] IACR 2017/1050
[2] IACR 2017/1066
[3] IACR 2018/046

The goal

State-of-the-art research seeks to produce **general proving systems** with **no trust requirements** for soundness and **practical efficiency**.

Some interesting work uses a universal and/or updateable reference string, but this has subtle implications for the trust model.

It's trickier than it sounds, even if it already sounds tricky.

Thank you

Questions are welcome.

Transaction privacy and asset fungibility are tricky and unsolved, so please ask away.

`sarang.noether@protonmail.com`