Berkeley CS276 & MIT 6.875

Zerocash:
addressing Bitcoin's privacy problem

Zcash

Lecturer: Raluca Ada Popa
Nov 3, 2020

Slides from Prof. Alessandro Chiesa

Recording..

Today

Using **zero-knowledge proofs** (and commitments and PRFs) from previous lectures

to design

Zerocash, a privacy-preserving cryptocurrency (private version of Bitcoin), a real system with almost $1 billion market cap

Zcash

- We will focus on the protocol design and use ZKs as a black box

Transactions In Bitcoin

(simplified)

From	Alice
To	Bob
Amount	3
Auth	s_{Alice}

From	Scrooge
To	Donald
Amount	5
Auth	$s_{Scrooge}$

…	…
…	…
…	…

From	Bob
To	Eve
Amount	3
Auth	s_{Bob}

Every payment transaction reveals:
1. the sender,
2. the receiver,
3. the amount.

This should raise some worries!

Payment History Reveals Lots

- medical information (specialty of your doctors)

  ➡ insurance companies could use it to increase premium or even deny coverage

- current and past locations (your travel patterns)

  ➡ gold mine for stalkers, burglars, assassins, …

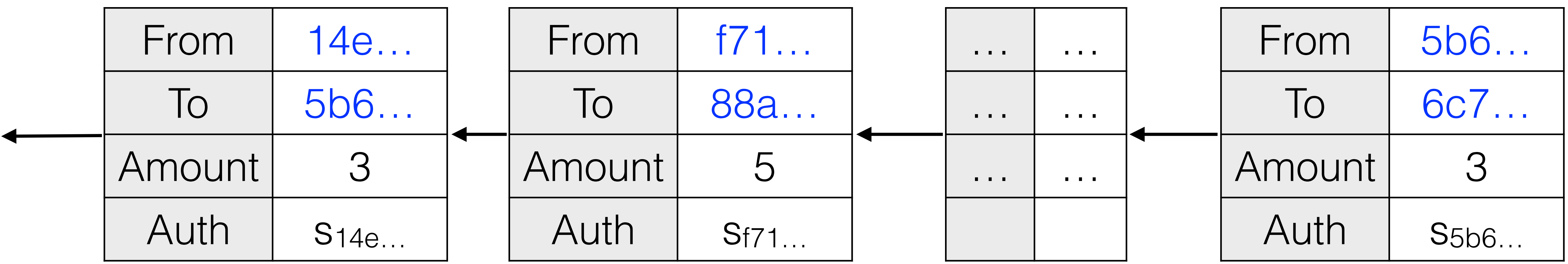
- merchant cash flow (suppliers, daily sales, …)

  ➡ intel for competitors

Compare:
GLBA (*Gramm-Leach-Bliley Act*) requires financial institutions to explain their info-sharing practices and safeguard sensitive and financial data. Each violation incurs civil penalties of up to $100K.

Transactions In Bitcoin

(simplified)

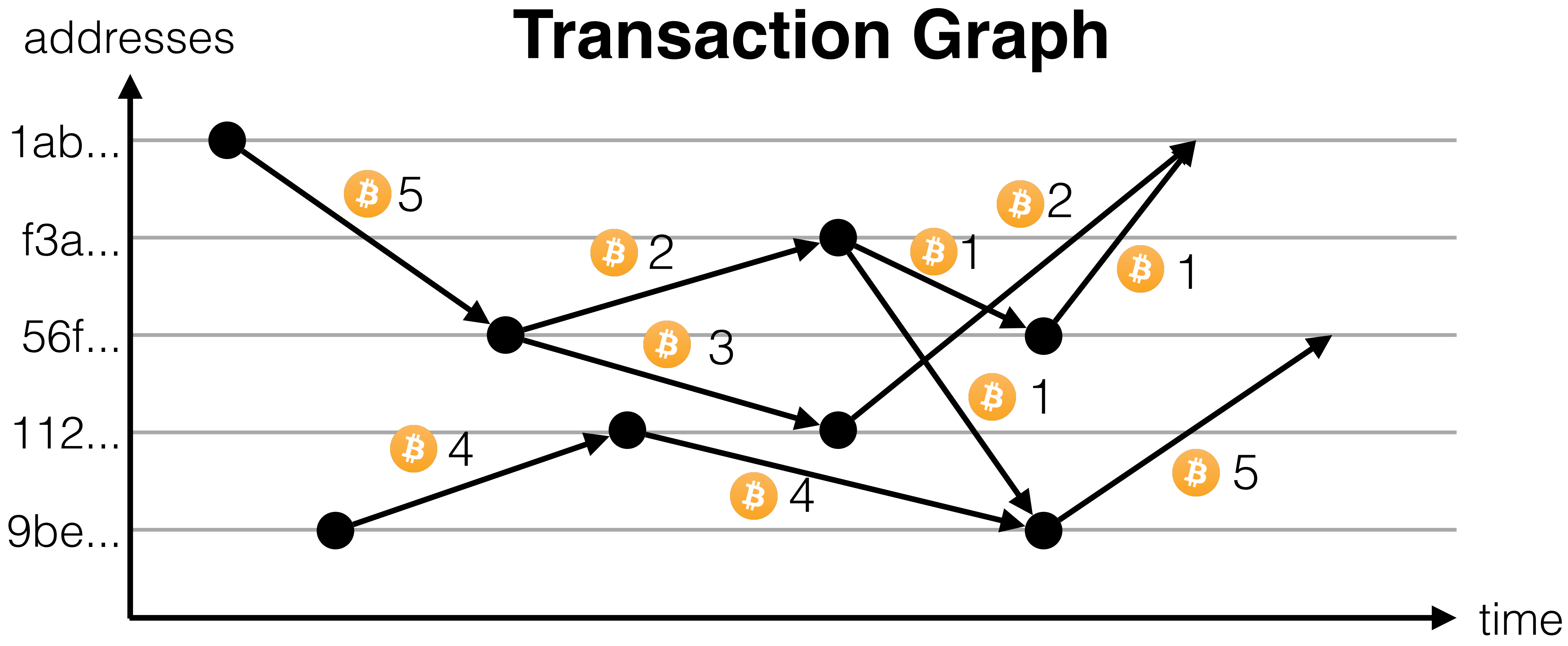


"But there are no names. Those are just addresses!"

"Those are just addresses."

These are known by everyone you interact with.

And literally anyone can analyze the ledger.



transaction graph + side-info → addresses become names of people!

Not just theoretical:

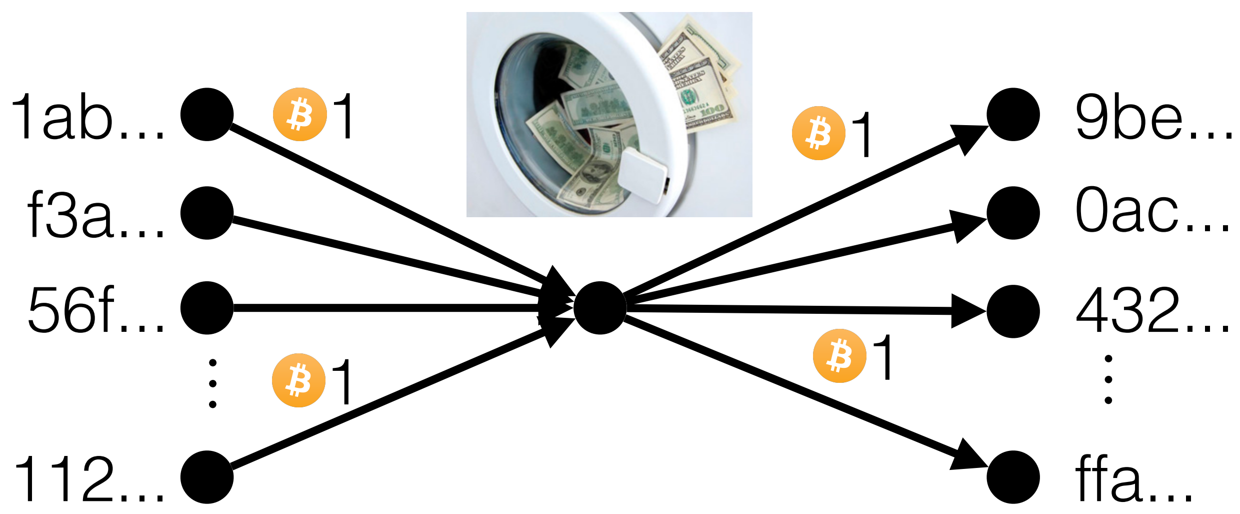
FBI Silk Road investigations, IRS subpoena to Coinbase, deanon studies, …

[Reid Martin 11] [Barber Boyen Shi Uzun 12] [Ron Shamir 12] [Ron Shamir 13]
[Meiklejohn Pomarole Jordan Levchenko McCoy Voelker Savage 13] [Ron Shamir 14]

Mitigations

Use new address for each payment.

Launder money with others.



"Seems" harder to analyze.

But tracks remain…

and recent quantitative results exploit such tracks. [MMLN17] [KFTS17]

Bitcoin history is publicly stored **forever**.

Methods of analysis only get **stronger**.

Fungibility

a dollar is a dollar, regardless of its history

Recognized as crucial property of money 350+ years ago.

(*Crawfurd v. The Royal Bank*, 1749)

Bitcoin is **NOT** fungible
because a coin's pedigree is public.

In particular, a coin's value is ill-defined:

- different people value the same coin differently
- the same person values different coins differently
- heuristic: new coins more valuable than old ones
- central party that determines correct value?

If privacy is so important why isn't Bitcoin private?

From	Alice	From	Scrooge	…	…	From	Bob
To	Bob	To	Donald	…	…	To	Eve
Amount	3	Amount	5	…	…	Amount	3

How does the world know that Bob has 1 Bitcoin to spend?

check that he received it, and that he did not spend it

What if users encrypted their payment transactions?

From	**Enc(**A**)**	From	**Enc(**S**)**	…	…	From	**Enc(**B**)**
To	**Enc(**B**)**	To	**Enc(**D**)**	…	…	To	**Enc(**E**)**
Amount	**Enc(**3**)**	Amount	**Enc(**5**)**	…	…	Amount	**Enc(**3**)**

Not clear how to check a payment's validity.

privacy and accountability are at odds

Zerocash

A cryptographic protocol achieving a digital currency that is:

Decentralized

works on any append-only ledger

Privacy-preserving

anyone can post a payment transaction to anyone else,
while provably hiding the payment's sender, receiver, amount

Efficient

payment transactions take few seconds to produce,
are less than 1KB in size, and take a few milliseconds to verify

The Basic Intuition

From	**Enc**(A)
To	**Enc**(B)
Amount	**Enc**(1)
Proof	π

From	**Enc**(S)
To	**Enc**(D)
Amount	**Enc**(2)
Proof	π'

From	**Enc**(B)
To	**Enc**(E)
Amount	**Enc**(1)
Proof	π''

From	c_1
To	c_2
Amount	c_3
Proof	π'''

I am publishing three ciphertexts c_1, c_2, c_3.

They contain the encryptions of a sender address,
a receiver address, and a transfer amount respectively.

Moreover, the amount transfered has not been double spent.

I have generated a cryptographic proof π''' that all of this is true.

Q1: what kind of cryptographic proof?

Q2: what exactly is the statement being proved?

A Little Beyond Intuition

From	**Enc(**A**)**
To	**Enc(**B**)**
Amount	**Enc(**1**)**
Proof	π

From	**Enc(**S**)**
To	**Enc(**D**)**
Amount	**Enc(**2**)**
Proof	π'

From	**Enc(**B**)**
To	**Enc(**E**)**
Amount	**Enc(**1**)**
Proof	π''

Q1: what kind of cryptographic proof?

zero knowledge (nothing revealed beyond truth of statement)

succinct (proof is very short and cheap to verify)

non-interactive (need to write it down!)

proof (true statements have proofs, false ones do not)

of knowledge (technical… allows using crypto in statement)

ZK-SNARK

There are efficient constructions

`libsnark.org`

A Little Beyond Intuition

From	**Enc**(A)
To	**Enc**(B)
Amount	**Enc**(1)
Proof	π

From	**Enc**(S)
To	**Enc**(D)
Amount	**Enc**(2)
Proof	π'

From	**Enc**(B)
To	**Enc**(E)
Amount	**Enc**(1)
Proof	π''

Q2: what exactly is the statement being proved?

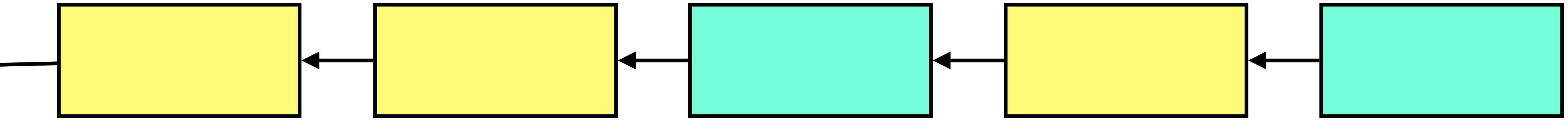
This is not trivial. Let's have some design fun.

Goals - recap

Only owner of a coin can spend it
.. and cannot double spend it

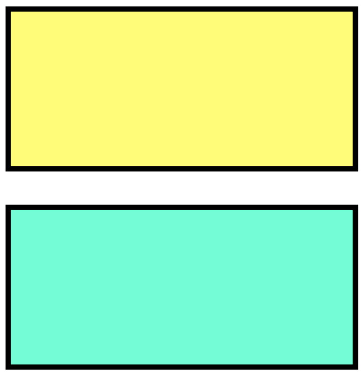
No PPT adversary can link transactions to sender or receiver or learn amounts

Attempt #0: template



view of blockchain

Transaction types

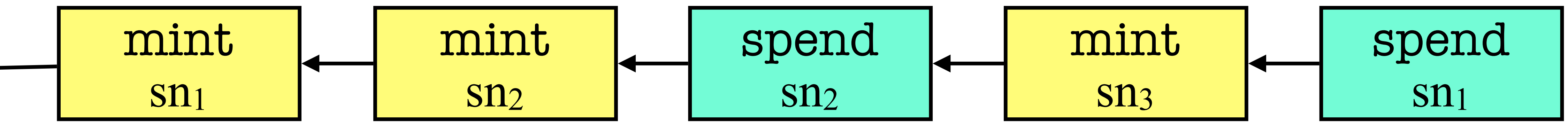


type 1

type 2

coin

Attempt #1: plain serial numbers

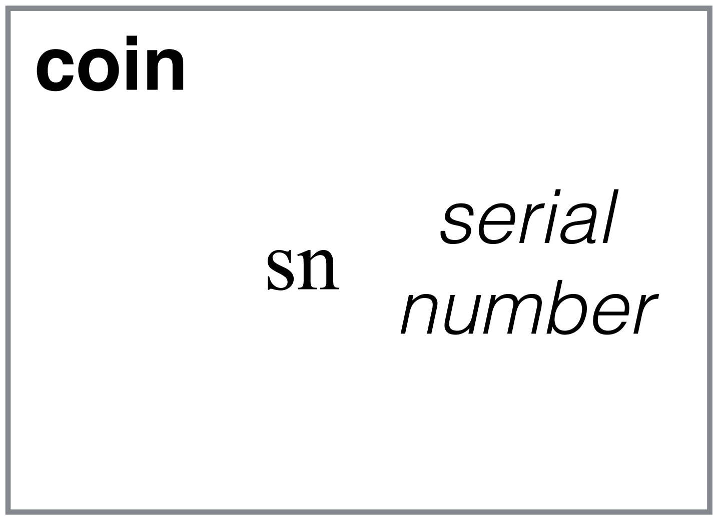


view of blockchain

Transaction types

mint sn Consume 1 BTC to create a value-1 coin w/ serial number sn.

spend sn Consume the coin w/ serial number sn.



Good:

cannot double spend

Bad:

spend linkable to its mint

anyone can spend!

…

Recall: Commitment Scheme

A commitment protocol is an efficient two-stage protocol between a sender S and a receiver R:

- commitment stage: S has private input *x*. At the end of the stage,
  - Both parties hold *com* (commitment)
  - S holds *r* (the randomness used for recommitment)
- reveal stage: S sends *(r, x)* to R, which accepts or rejects

Completeness: R always accepts in an honest execution of S.

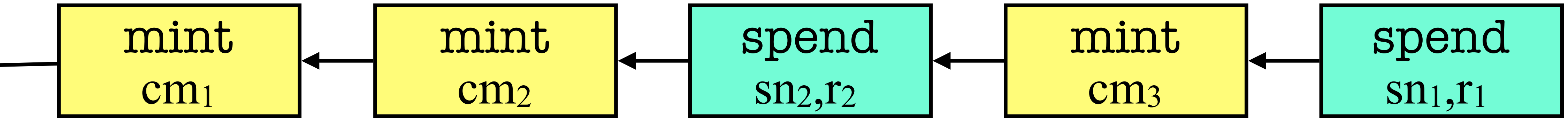
Hiding: Hiding:∀ R*, $x \neq x'$, in commit stage

$$\{ \text{View}(S(x), R^*)(1^k) \} \approx_c \{ \text{View}(S(x'), R^*)(1^k) \}.$$

Binding: Let *com* be output of commit stage, ∀S*

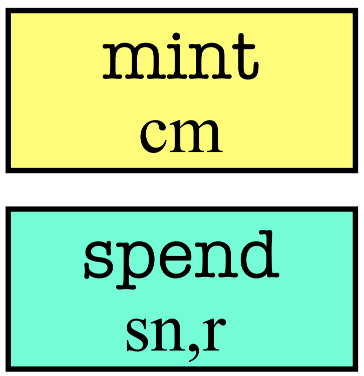
Prob[S* can reveal two pairs (r,x) & (r',x') s.t. R(com, r, x) = R(com, r′, x′) = Accept] <negl(k)

Attempt #2: committed serial numbers



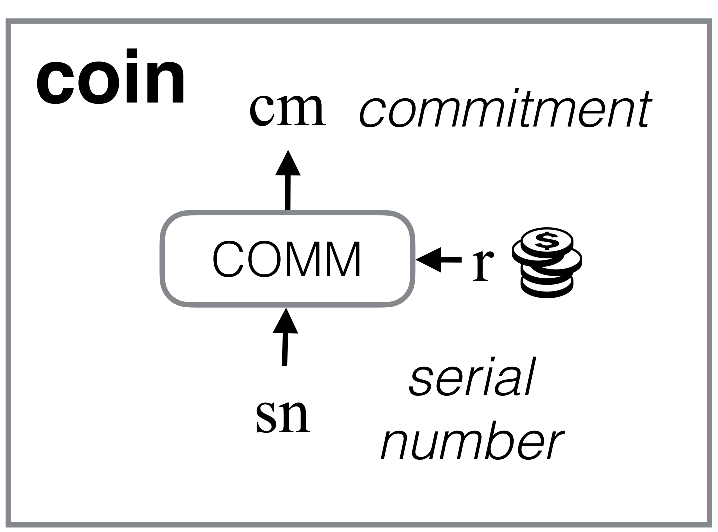
view of blockchain

Transaction types



Consume 1 BTC to create a value-1 coin w/ commitment cm.

Consume the coin w/ serial number sn.



Good:

cannot double spend

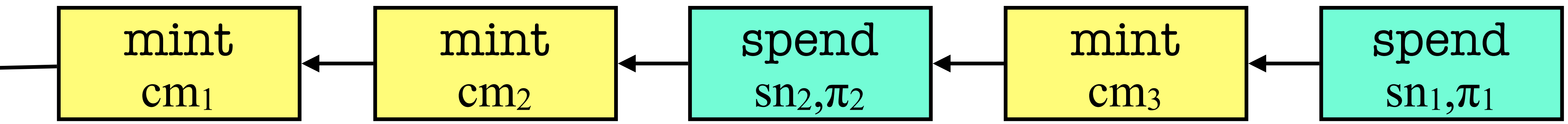
others can't spend my coins

Bad:

spend linkable to its mint

...

Attempt #3: ZKPoK of commitment



view of blockchain

Transaction types

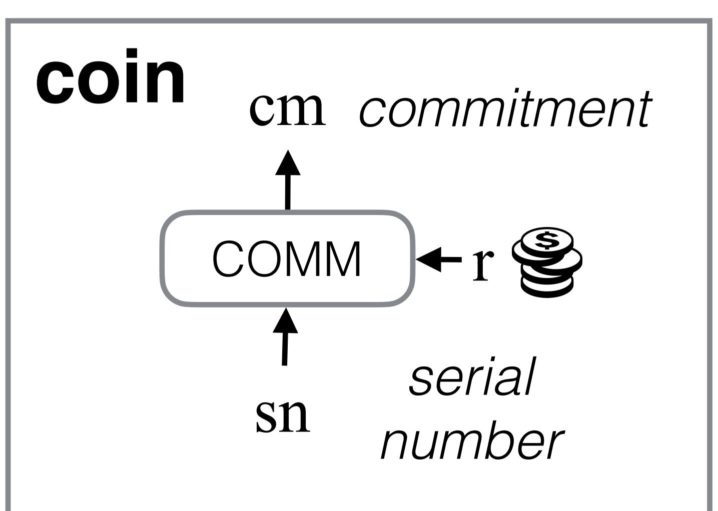
mint cm — Consume 1 BTC to create a value-1 coin w/ commitment cm.

spend sn,π — Consume the coin w/ serial number sn.

Here is a ZK proof π that I know secret r s.t.

- **exists** • $cm \in$ "list of prior commitments"
- **well-formed** • cm=COMM(sn; r)

In Zerocash, commitments are arranged in a Merkle tree and spender proves that it knows an authentication path from a leaf with cm to the public Merkle root



Good:

cannot double spend

others can't spend my coins

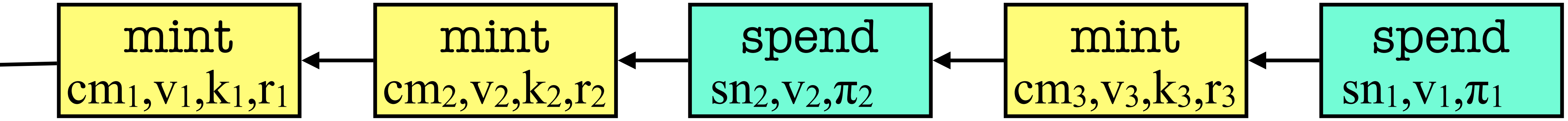
spend and mint unlinkable

Bad:

fixed denomination

…

Attempt #4: variable denomination



view of blockchain

It might look that cm is not needed because r is released in the mint. But the binding property of cm still holds which binds v and k. In Zerocash, cm is a leaf in the Merkle tree of commitments, enabling the spender to prove that cm is in the list of prior commitments. There are ways to implement the protocol without cm by putting different things in the Merkle tree or potentially a ZK proof at mint.

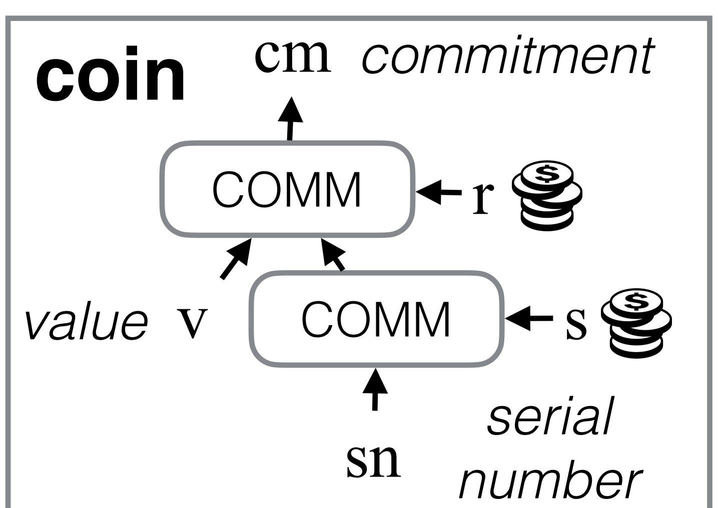
Transaction types

mint cm,v,k,r — Consume v BTC to create a value-v coin w/ commitment cm.

spend sn,v,π — Consume the value-v coin w/ serial number sn.

Here is a ZK proof π that I know secret (r,s) s.t.

- **exists** • $cm \in$ "list of prior commitments"
- **well-formed** • $cm=COMM(v,k; r)$ & $k=COMM(sn; s)$



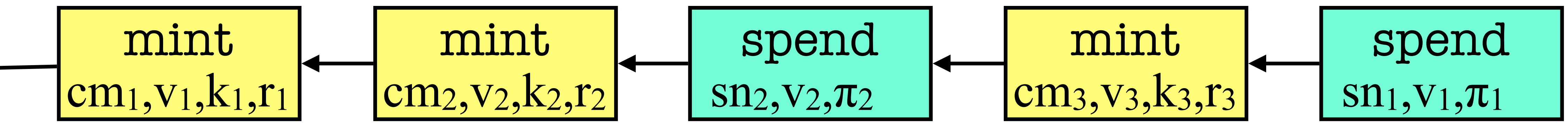
Good:

cannot double spend
others can't spend my coins
variable denomination

Bad:

spend and mint partially linkable
…

Attempt #5: payment addresses



view of blockchain

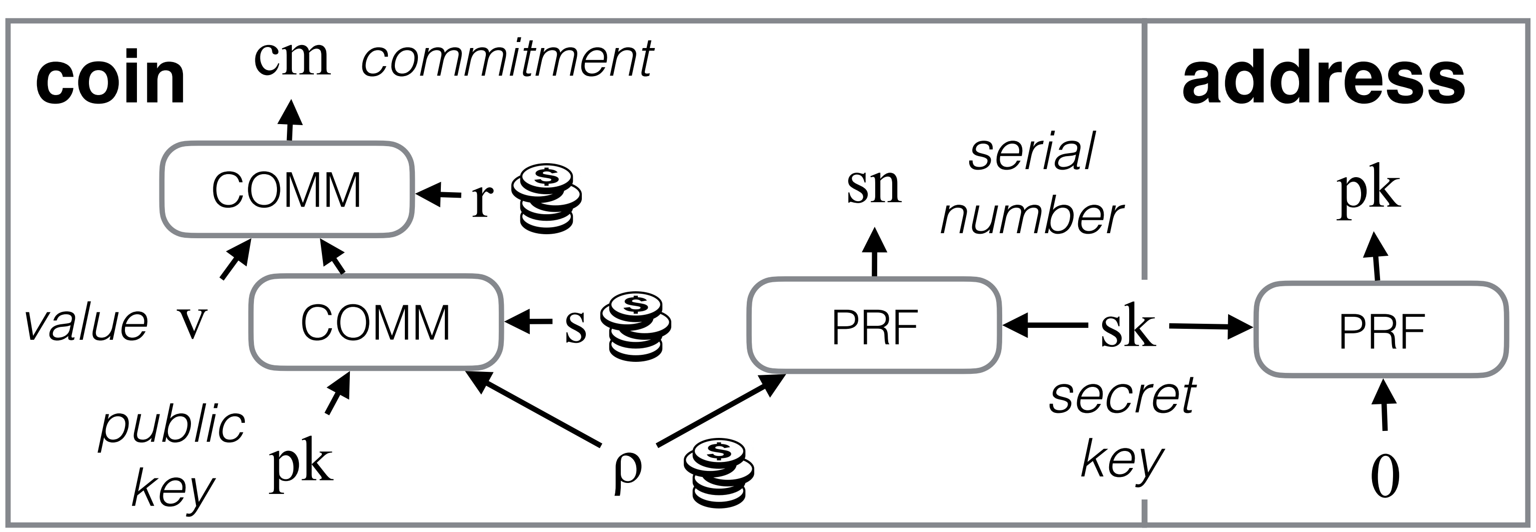
Transaction types

mint cm,v,k,r — Consume v BTC to create a value-v coin w/ commitment cm.

spend sn,v,π — Consume the value-v coin w/ serial number sn.

Here is a ZK proof π that I know secret (cm,k,r,s,ρ,pk,sk) s.t.

- **exists** • $cm \in$ "list of prior commitments"
- **well-formed** • $cm = \text{COMM}(v,k;\ r)$ & $k = \text{COMM}(pk,\rho;\ s)$
- **mine** • $sn = \text{PRF}(\rho;\ sk)$ & $pk = \text{PRF}(0;\ sk)$



Good:

cannot double spend

others can't spend my coins

spend and mint partially

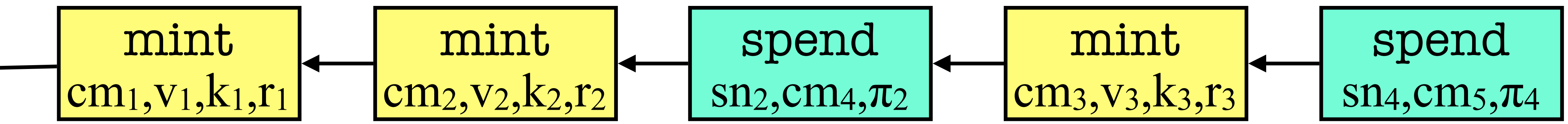
unlinkable

variable denomination

Bad:

reveals v

Attempt #6: direct payments



view of blockchain

Transaction types

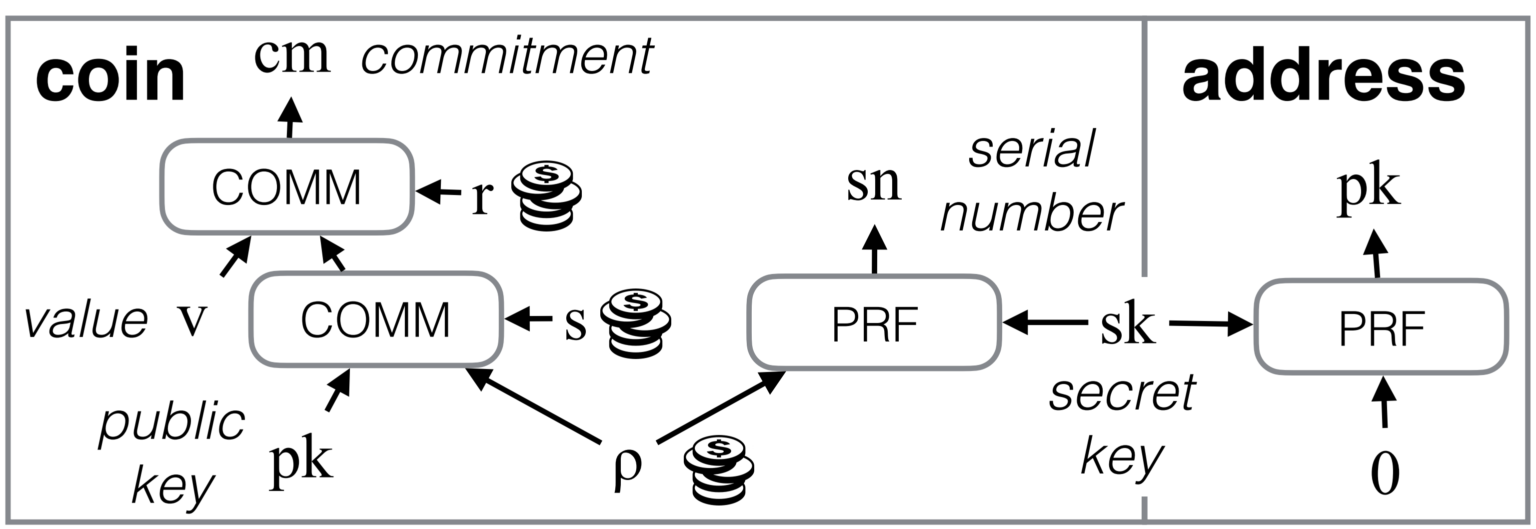
mint cm, v, k, r — Consume v BTC to create a value-v coin w/ commitment cm.

spend sn^A, cm^B, π — Consume coin w/ serial number sn^A & create coin w/ commitment cm^B.

Here is a ZK proof π that I know secret $(cm^A, v^A, k^A, r^A, s^A, \rho^A, pk^A, sk^A)$ s.t.

- **exists** • $cm^A \in$ "list of prior commitments"
- **well-formed** • $cm^A = \text{COMM}(v^A, k^A; r^A)$ & $k^A = \text{COMM}(pk^A, \rho^A; s^A)$
- **mine** • $sn^A = \text{PRF}(\rho^A; sk^A)$ & $pk^A = \text{PRF}(0; sk^A)$
- **well-formed** • $cm^B = \text{COMM}(v^B, k^B; r^B)$ & $k^B = \text{COMM}(pk^B, \rho^B; s^B)$
- **same value** • $v^A = v^B$

$(cm^B, v^B, k^B, r^B, s^B, \rho^B, pk^B)$ — send out-of-band or via blockchain

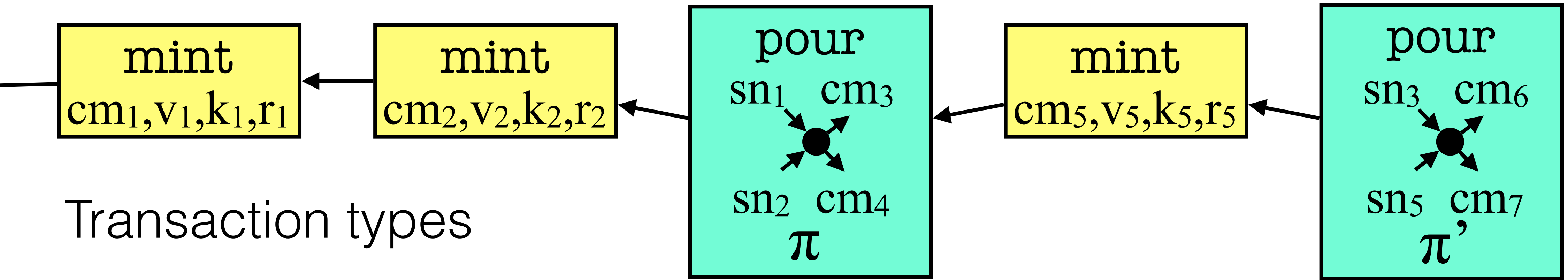


Good:

cannot double spend
others can't spend my coins
spend and mint unlinkable
variable denomination
hides sender, receiver, amt

Bad: join and split coins?

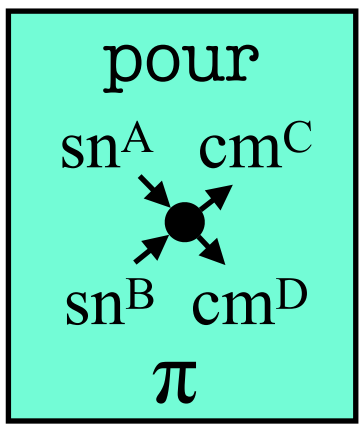
Sketch of Final Design



view of blockchain

Transaction types

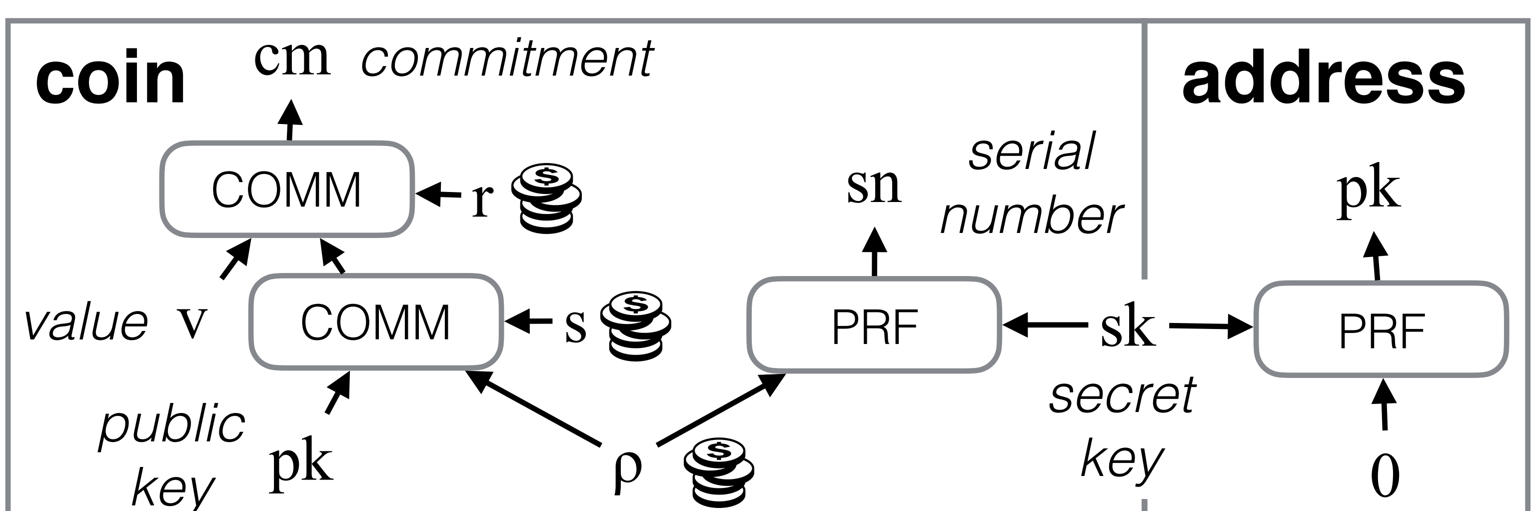
Consume v BTC to create a value-v coin w/ commitment cm.



Consume (my) **input** coins w/ serial numbers sn^A and sn^B in order to create two **output** coins (maybe not mine) w/ commitments cm^C and cm^D.

Here is a ZK proof π that I know secrets that demonstrate that

- the input coins were minted at some point in the past,
- the output coins are well-formed,
- balance is preserved.



Single tx type for:

- ✓ simple payments
- ✓ join coins
- ✓ split coins
- ✓ making change
- ✓ pay transaction fees

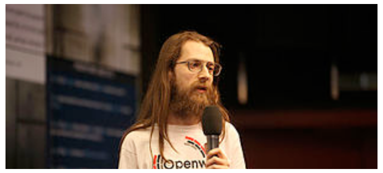
Academic Practical → Real-World Practical

2014.05: proof-of-concept implementation of *Zerocash*

2016.10: deployment of ZCASH

… 2+ years of research & development by startup (ZECC) to bridge the gap between academic implementation and a deployable system

- thourough analysis and vetting (even found a completeness bug! 😂)
- protocol changes
- efficiency improvements
- external security audits — nccgroup (freedom from doubt), Coinspect

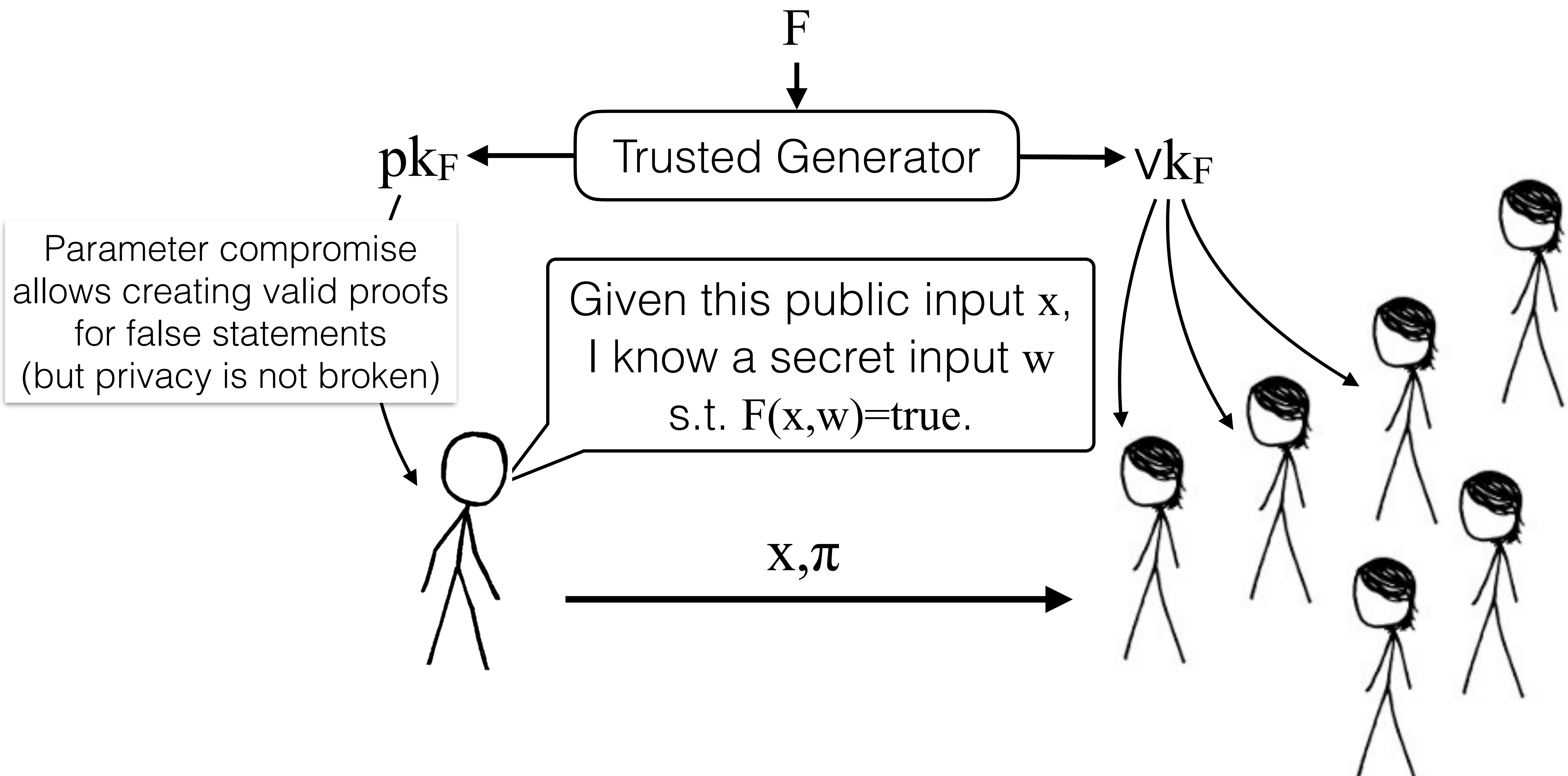


Solar Designer
(Alexander Peslyak)

- creation of clients, integration with wallets and exchanges
- **generation of public parameters for the ZK-SNARK (ZK proof system)**

The Pain of Public Parameters

Practical constructions of ZK-SNARKs need a trusted party to generate parameters for proving/verifying statements.



Who generates the parameters??

One approach: a set of people via a distributed multi-party protocol.

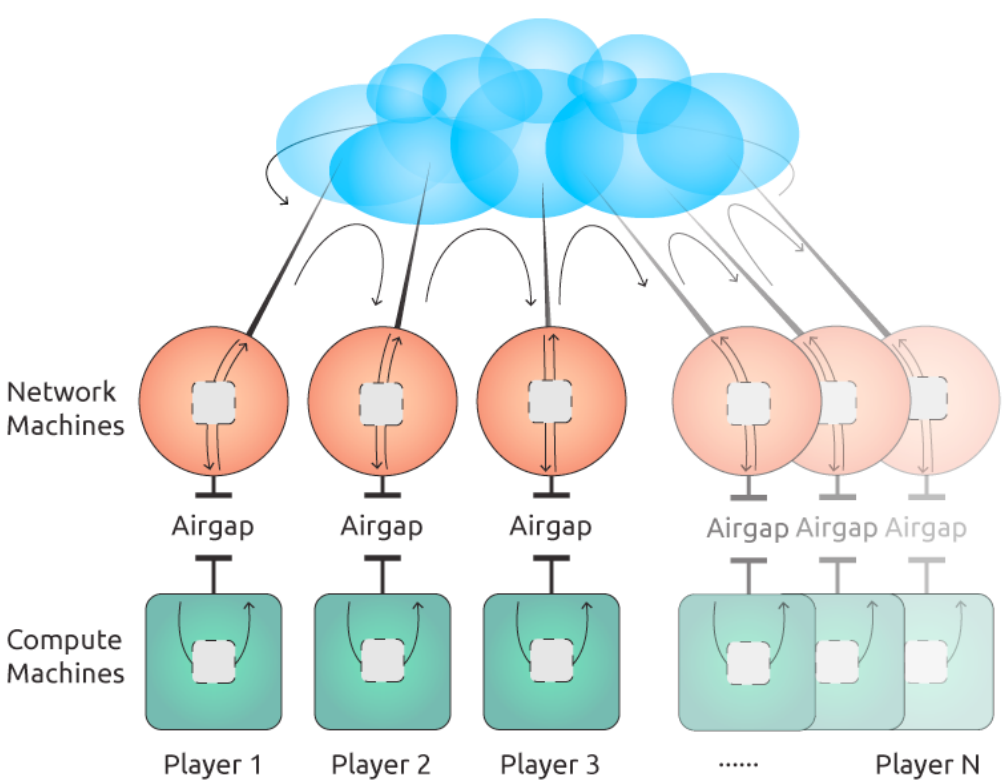
MPC Ceremony

Run by ZECC during October 22—23, 2016.

Main ingredients:

- n-party MPC protocol that is secure against ≤n-1 corruptions [B**C**GTV15][BGG16]
- extensive threat modeling and security engineering

airgap between network machines and compute machines



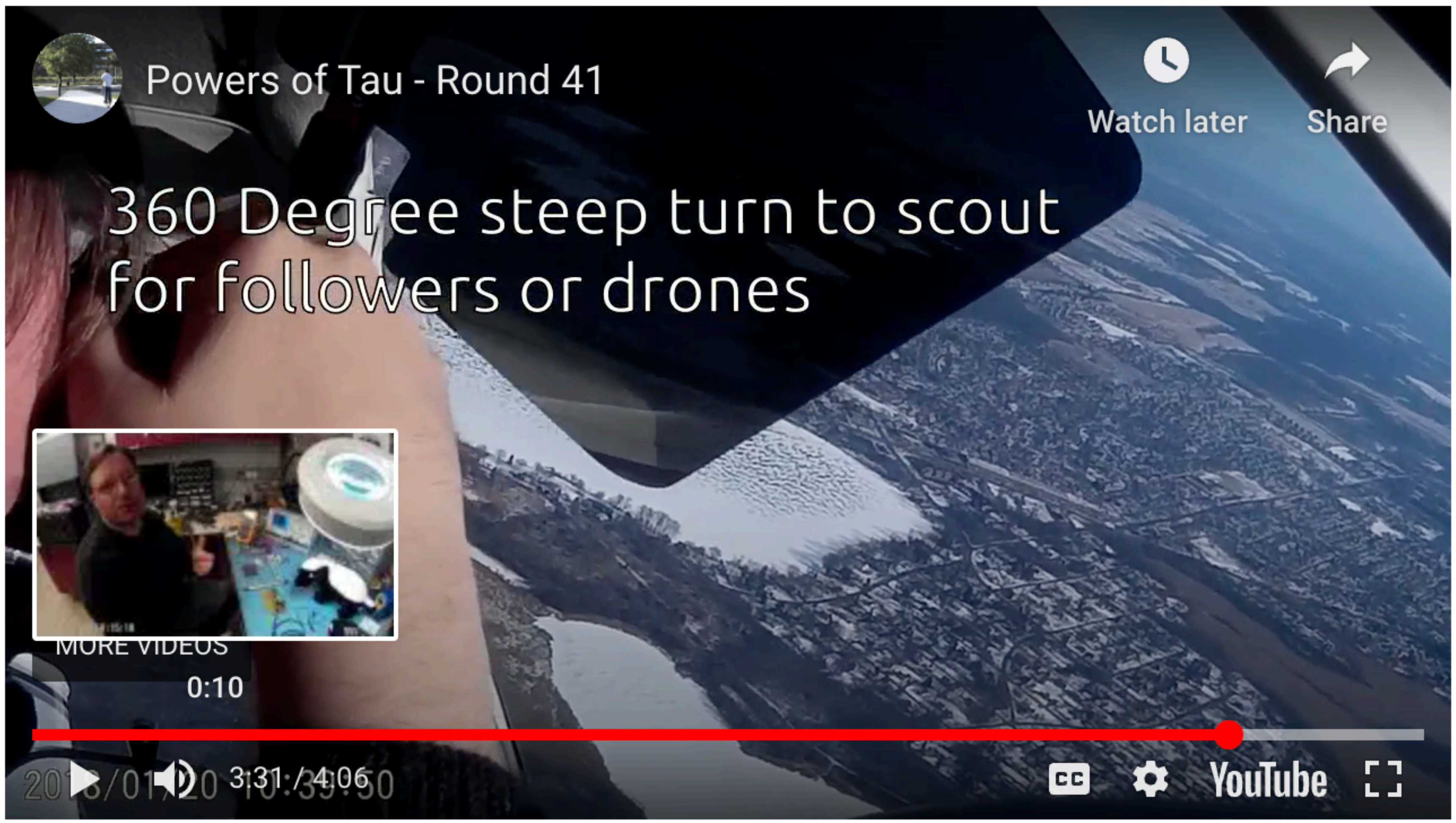
n=6 geographically distributed participants (including one security company, and a mobile station)

publicly-verifiable audit trail, in a hash chain stored on Twitter and the Internet Archive

video documentation from all participants including destruction of compute nodes

Some folks took randomness generation very seriously…

Using radioactive material from Chernobyl in an airplane…



Driving through the desert…

Some participants were hacked…

Cryptocurrency

https://explorer.zcha.in/

Zcash

Yesterday:

Price	Market Cap	Transactions	Total Monetary Base	Chain Height	Network Hashrate ?	Block Time	Difficulty
$52.73	$547,917,760	7,338,114	10,391,006 ZEC	1,028,961	5,638,593,480 Sol/s	76s	49,420,046

Zerocash/Zcash

https://explorer.zcha.in/

- uses zero-knowledge proofs to provide a privacy-preserving Bitcoin
- (I think) impressive use of advanced crypto like ZK proofs in practice, one of firsts
- secures almost a billion $