Zerocash: addressing Bitcoin's privacy problem

Alessandro Chiesa
UC Berkeley

Bitcoin's Privacy Problem

Would you like a new credit card?
You will pay almost no fees!

Sure! Any fine print?

We will publicly broadcast
every payment that you make.

Sender	Recipient	Amount	Time
Alice	Starbucks	$8.75	2017.06.02 @ 10:05
Alice	Uber	$11.50	2017.06.02 @ 11:00
…	…	…	…

No big deal.

Payment history reveals **lots** of information:

- medical information (specialty of your doctors)

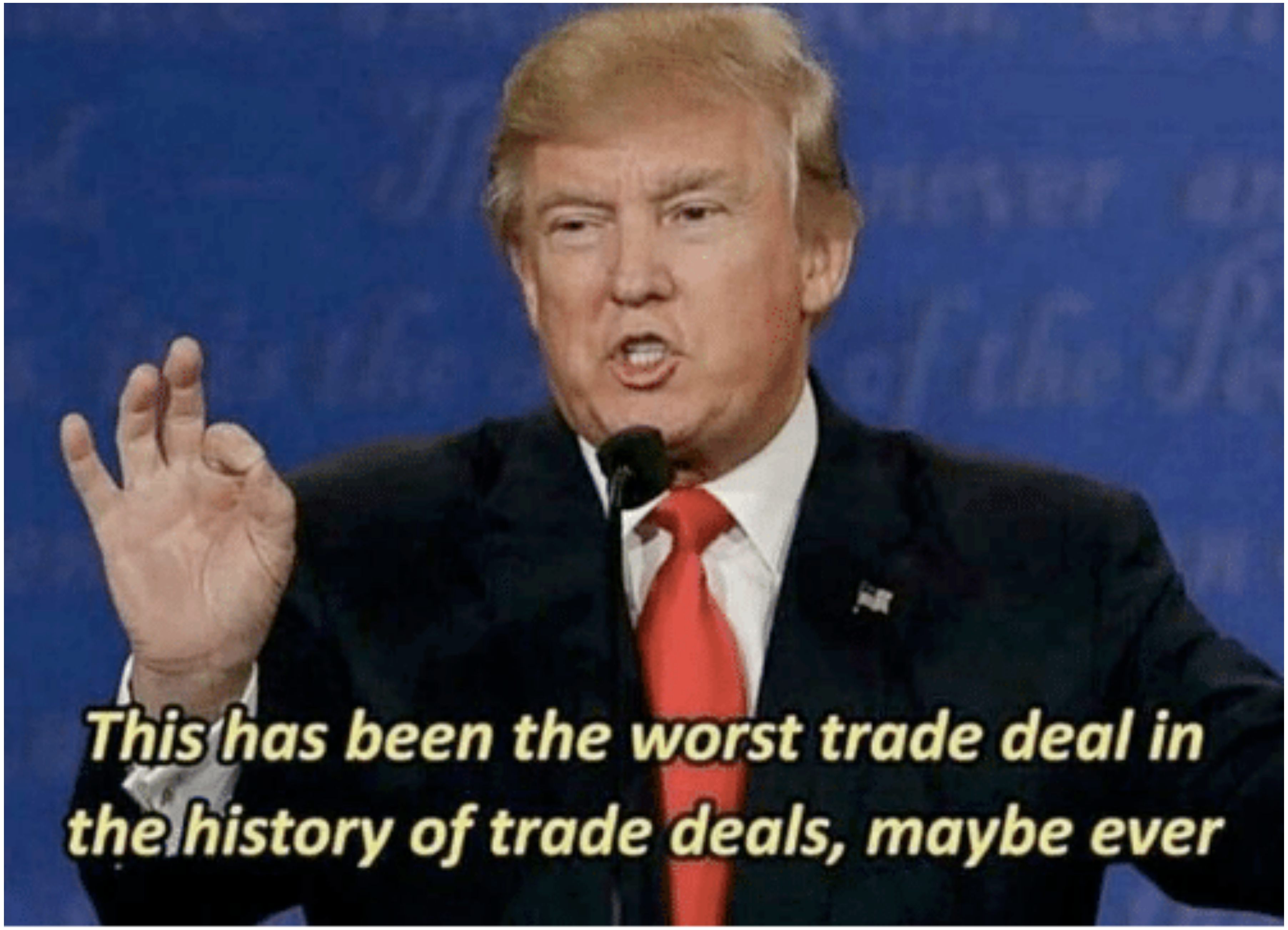
  ➡ insurance companies could use it to increase premium or even deny coverage

- current and past locations (your travel patterns)

  ➡ gold mine for stalkers, burglars, assassins, …

- merchant cash flow

  ➡ suppliers, daily sales, … all exposed to competitors


This has been the worst trade deal in
the history of trade deals, maybe ever

Your bank will not offer you this absurd deal.

Not just out of magnanimity:

Federal privacy laws **mandate** opt-out from data sharing.

GLBA (*Gramm-Leach-Bliley Act*) mandates
civil penalties of up to $100K per violation

What about Bitcoin?

no opt-out

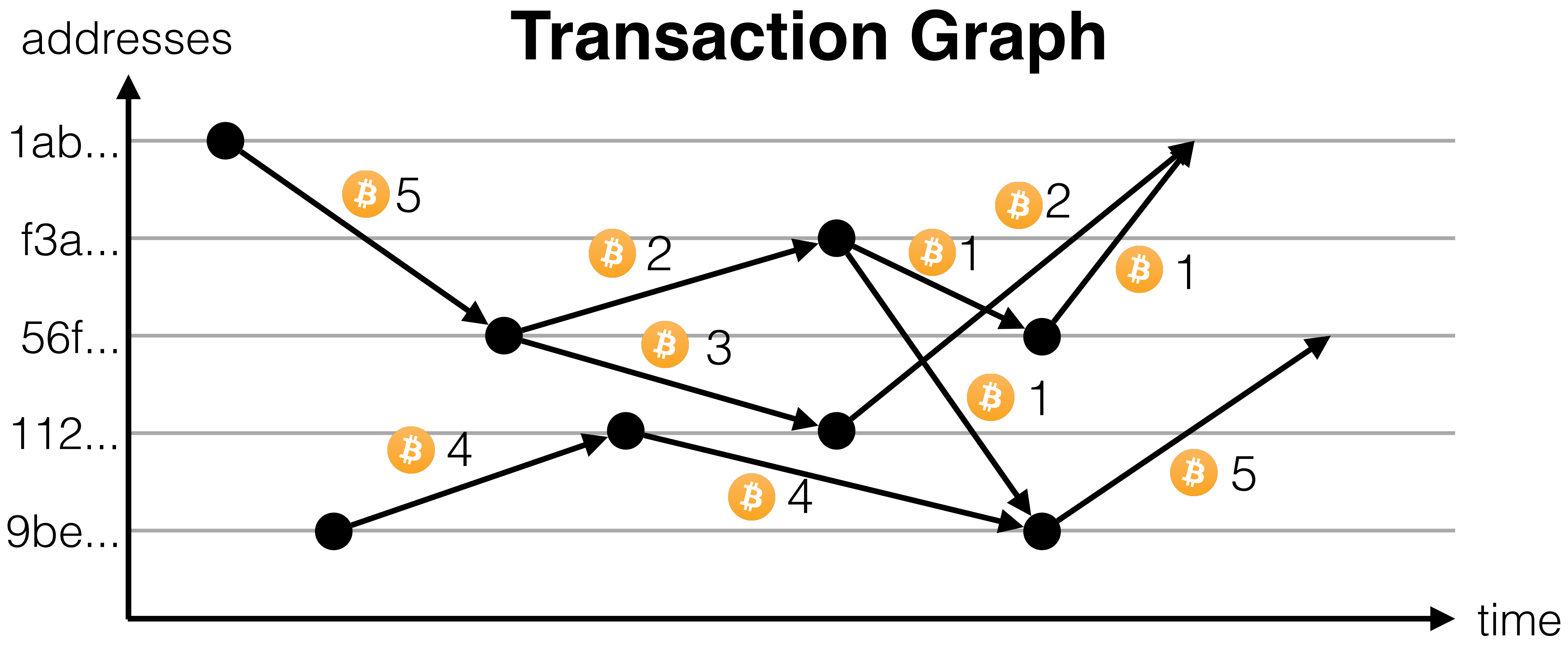
Sender	Recipient	Amount	Time
14e…	5b6…	₿ 8.75	2017.06.02@10:05
f71…	88a…	₿ 11.5	2017.06.02@11:00
…	…	…	…

"Not the same. These 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:

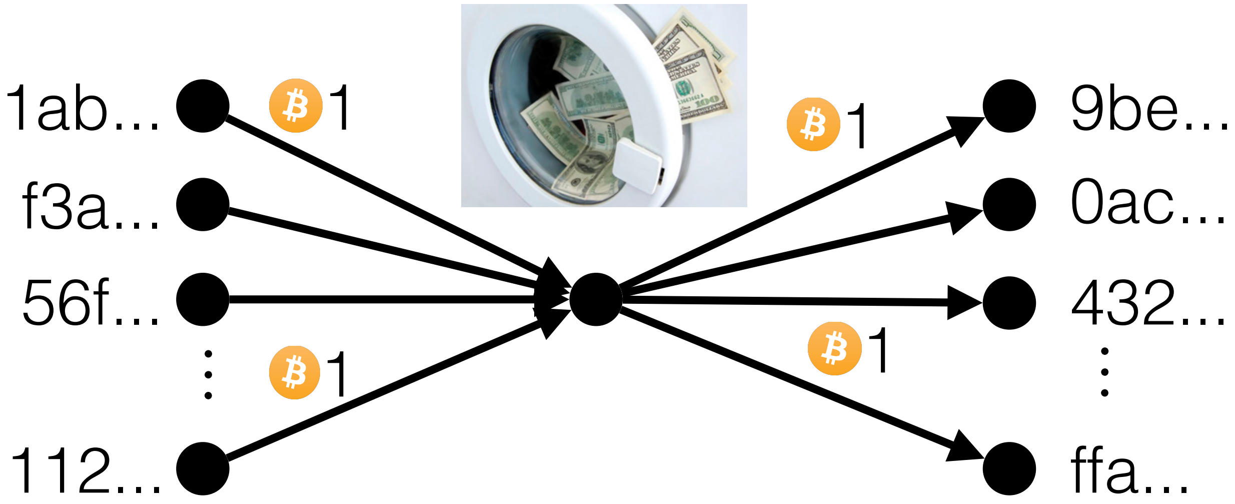
FBI Silk Road investigations, IRS subpoena to Coinbase, deanon studies, …

[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 to the Privacy Problem

Use new address for each payment.

Launder money with others.



"Seems" harder to analyze.

But tracks remain…

Recent quantitative results exploiting 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)

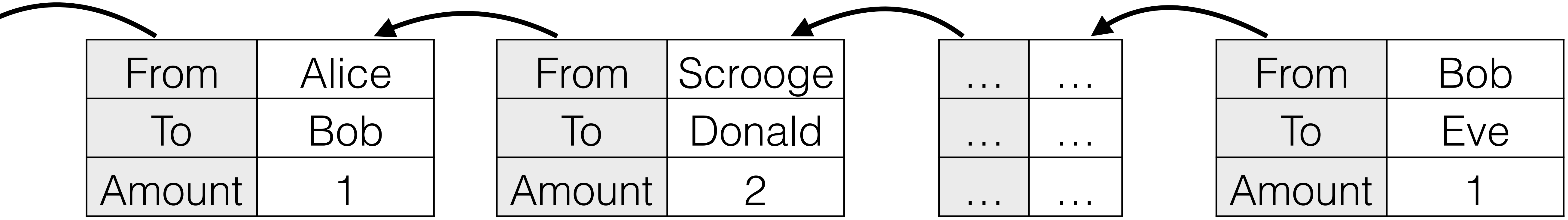
Bitcoin & co are **NOT** fungible
because a coin's pedigree is public.

Dangerous consequences:

- ill-defined value
    - 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?
- price discrimination (salary raise → rent hike)
- censorship (miners filter transactions)

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

Privacy vs Accountability



From	Alice
To	Bob
Amount	1

From	Scrooge
To	Donald
Amount	2

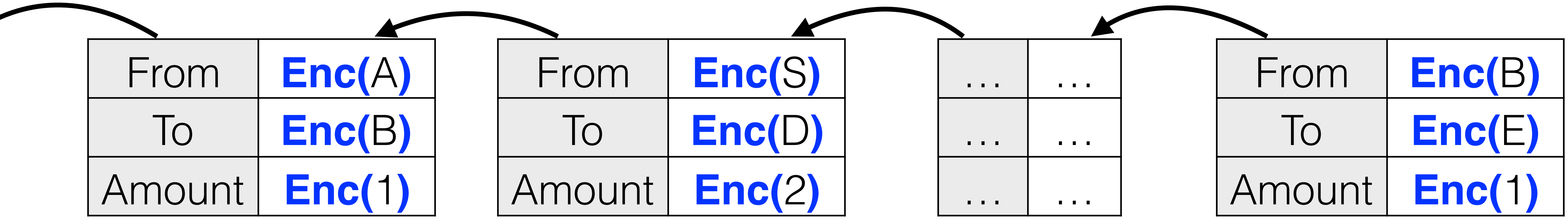
…	…
…	…
…	…

From	Bob
To	Eve
Amount	1

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)
To	**Enc**(B)
Amount	**Enc**(1)

From	**Enc**(S)
To	**Enc**(D)
Amount	**Enc**(2)

…	…
…	…
…	…

From	**Enc**(B)
To	**Enc**(E)
Amount	**Enc**(1)

Not clear how to check a payment's validity.

privacy and accountability are at odds

The Zerocash Protocol

Zerocash

A cryptographic protocol achieving a digital currency that is:

Decentralized

works when given any (ideal) ledger

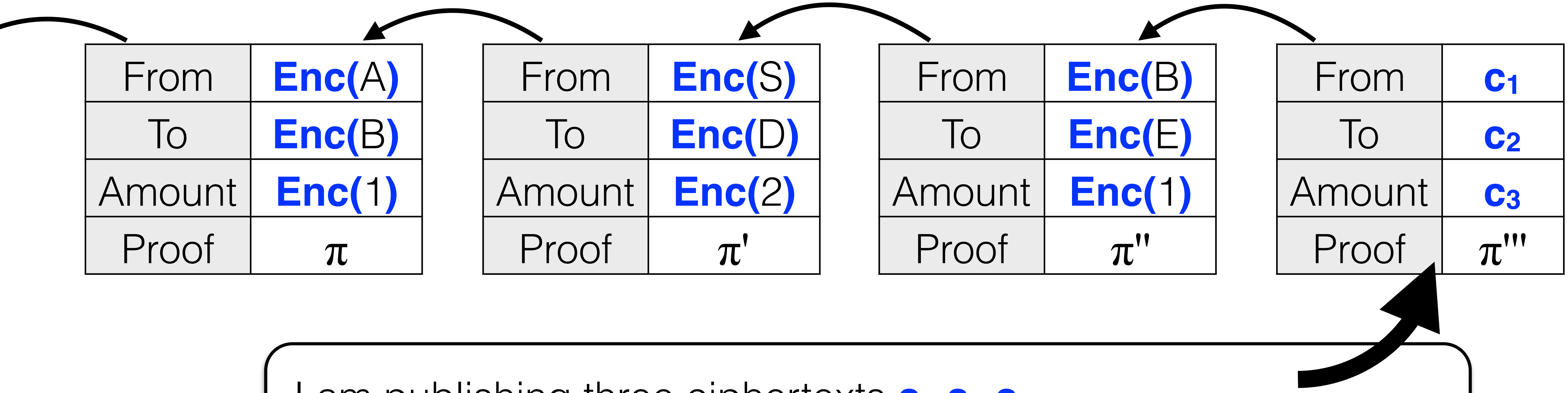
Privacy-preserving

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

Efficient

payment transactions take less than 1min 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.

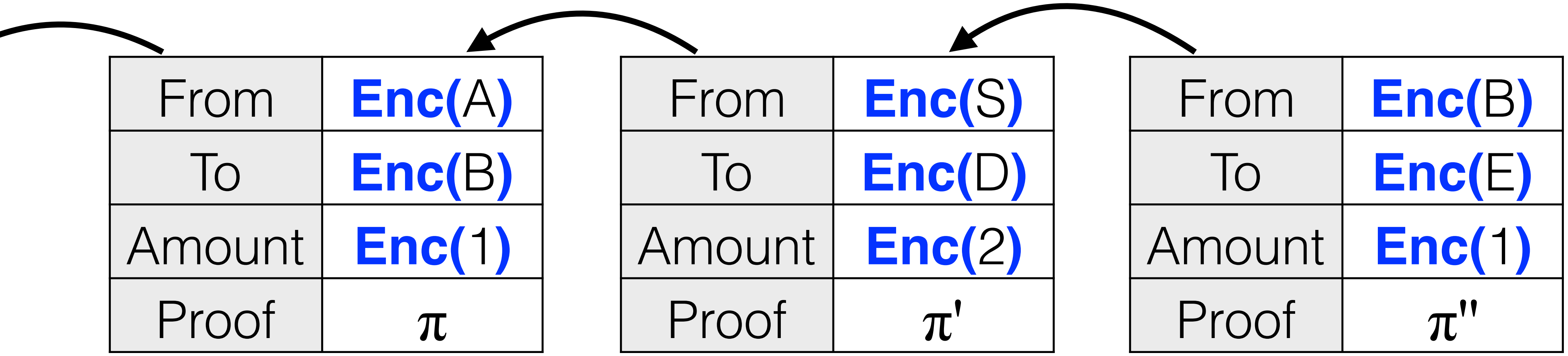
Moreover, the amount transfered has not been double spent.

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

Q1: what kind of crypto proof?

Q2: what exactly is the statement being proved?

Requirements on Crypto Proof

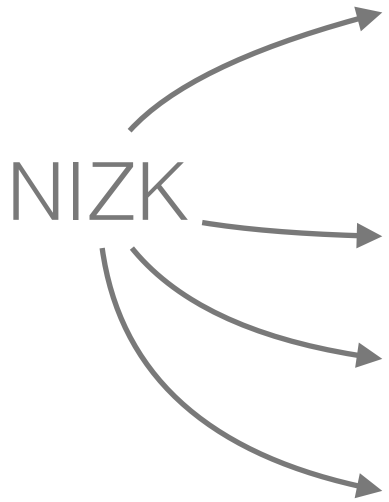


From	Enc(A)
To	Enc(B)
Amount	Enc(1)
Proof	π

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 crypto 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!)

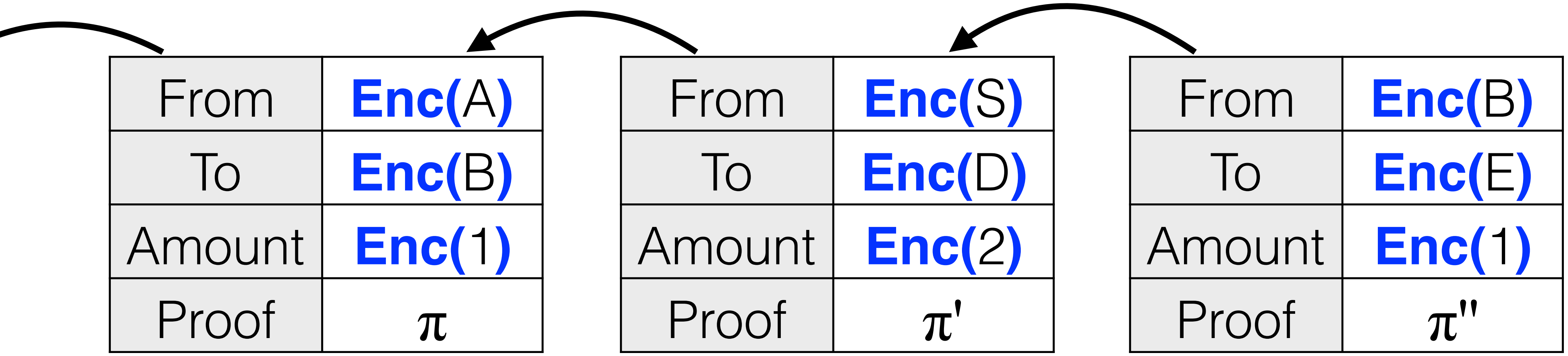
~~proof~~ argument (true statements have proofs, false ones do not)

of knowledge (technical… allows using crypto in statement)

ZK-SNARK have concretely efficient constructions

`libsnark.org`

Requirements on Crypto Proof



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

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

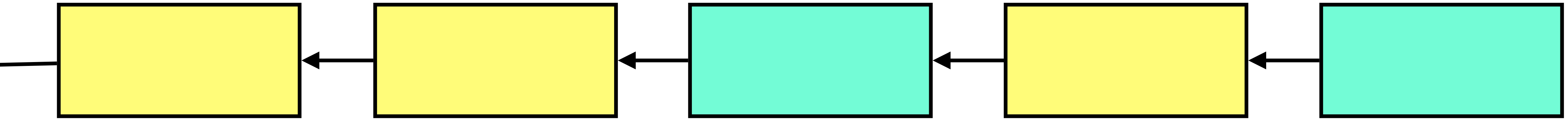
From	Enc(B)
To	Enc(E)
Amount	Enc(1)
Proof	π''

Q2: what exactly is the statement being proved?

this requires some thought

time to have some design fun

Attempt #0: template


view of blockchain

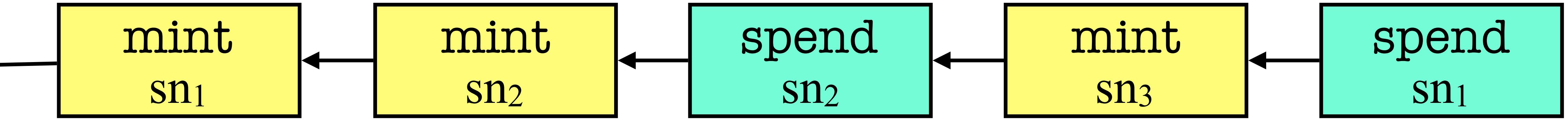
Transaction types

🟨 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.

coin

sn *serial number*

Good:

cannot double spend

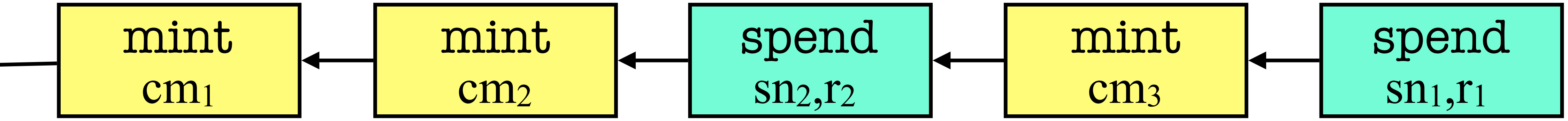
Bad:

spend linkable to its mint

anyone can spend!

…

Attempt #2: committed serial numbers



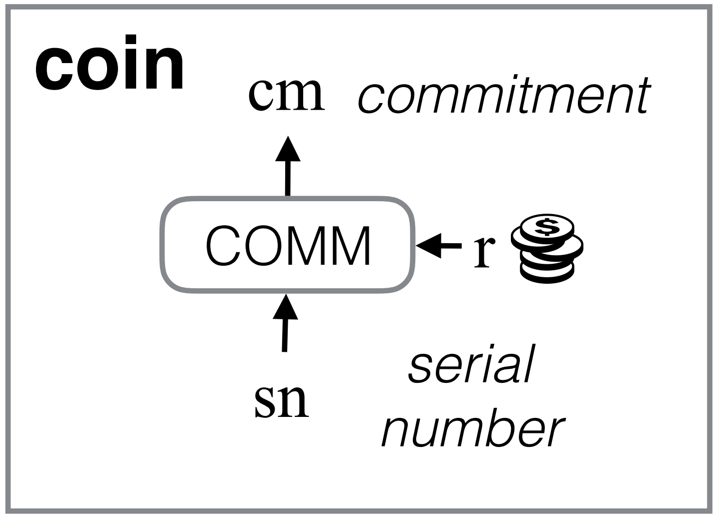
mint cm_1 ← mint cm_2 ← spend $\mathrm{sn}_2,\mathrm{r}_2$ ← mint cm_3 ← spend $\mathrm{sn}_1,\mathrm{r}_1$

view of blockchain

Transaction types

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

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



coin

cm *commitment*

COMM ← r

sn *serial number*

Good:

cannot double spend

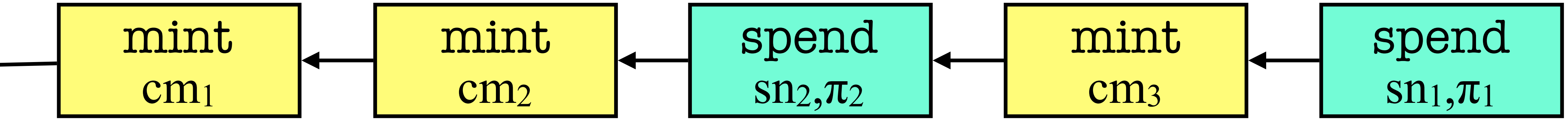
others can't spend my coins

Bad:

spend linkable to its mint

…

Attempt #3: ZKPoK of commitment



mint cm_1 ← mint cm_2 ← spend sn_2,π_2 ← mint cm_3 ← spend sn_1,π_1

view of blockchain

Transaction types

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

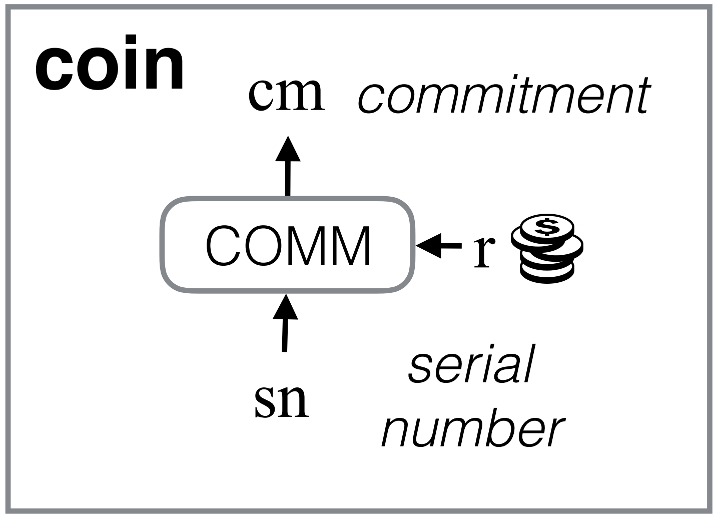
spend sn,π — Consume the coin w/ serial number sn.

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

[Sander Ta-Shma CRYPTO 1999]

exists • cm ∈ "list of prior commitments"

well-formed • cm=COMM(sn;r)



coin

Good:

cannot double spend

others can't spend my coins

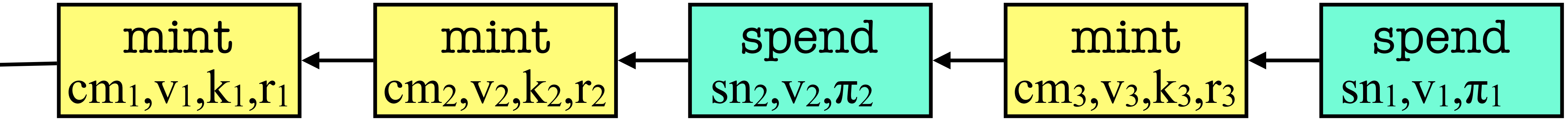
spend and mint unlinkable

Bad:

fixed denomination

…

Attempt #4: variable denomination



mint cm_1,v_1,k_1,r_1 ← mint cm_2,v_2,k_2,r_2 ← spend sn_2,v_2,π_2 ← mint cm_3,v_3,k_3,r_3 ← spend sn_1,v_1,π_1

view of blockchain

Transaction types

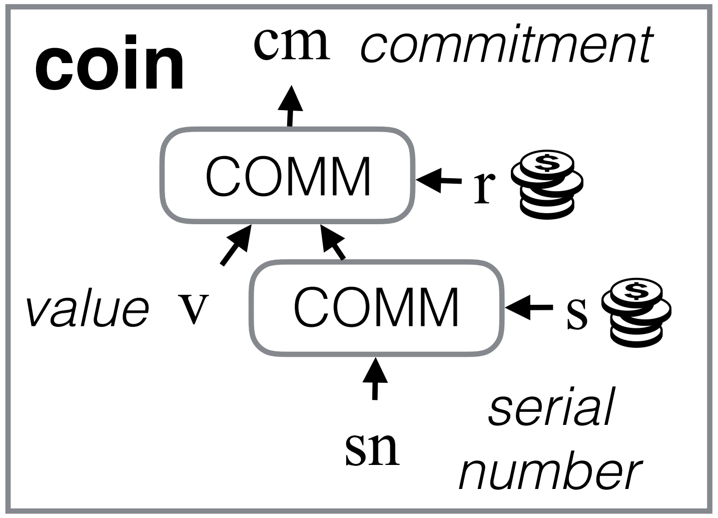
mint cm,v,k,r — Consume v BTC to create a value-v coin w/ commitment cm.

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

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

exists • $cm \in$ "list of prior commitments"

well-formed • $cm=$COMM$(v,k;r)$ & $k=$COMM$(sn;s)$



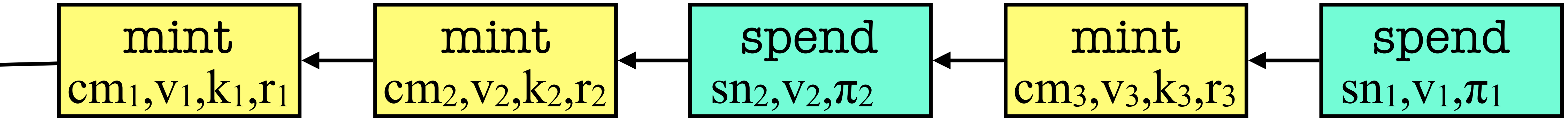
Good:

cannot double spend
others can't spend my coins
spend and mint unlinkable
variable denomination

Bad:

only hides sender
…

Attempt #5: payment addresses


mint
$\mathrm{cm}_1, \mathrm{v}_1, \mathrm{k}_1, \mathrm{r}_1$
mint
$\mathrm{cm}_2, \mathrm{v}_2, \mathrm{k}_2, \mathrm{r}_2$
spend
$\mathrm{sn}_2, \mathrm{v}_2, \pi_2$
mint
$\mathrm{cm}_3, \mathrm{v}_3, \mathrm{k}_3, \mathrm{r}_3$
spend
$\mathrm{sn}_1, \mathrm{v}_1, \pi_1$

view of blockchain

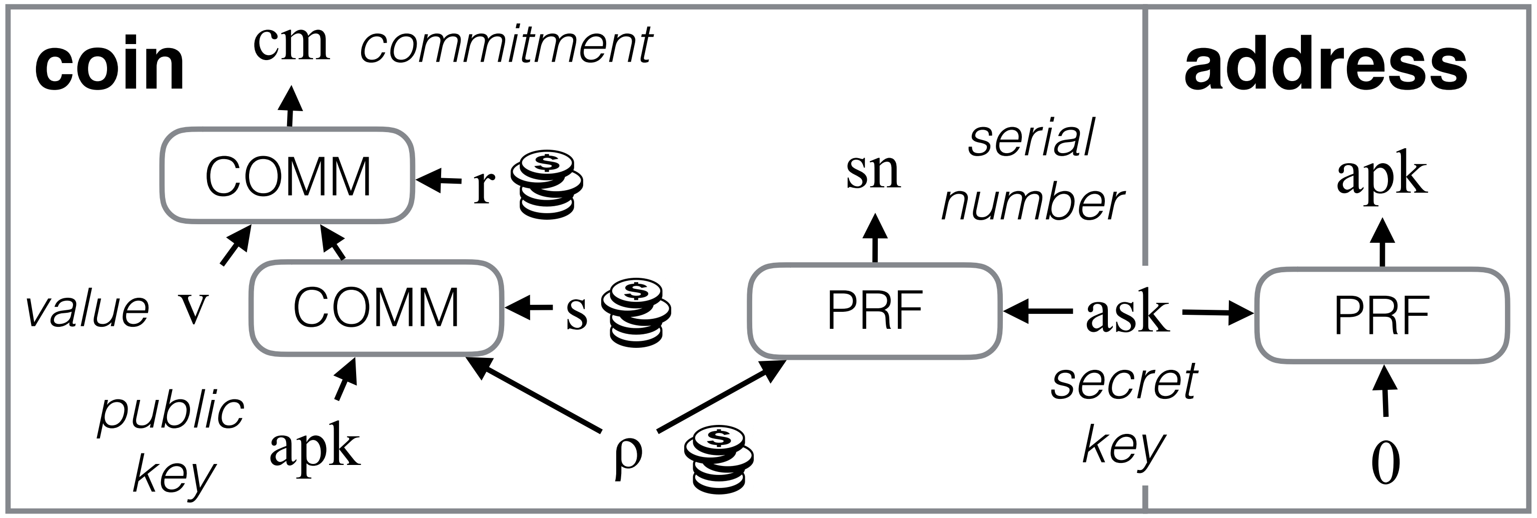
Transaction types

mint $\mathrm{cm}, \mathrm{v}, \mathrm{k}, \mathrm{r}$ — Consume v BTC to create a value-v coin w/ commitment cm.

spend $\mathrm{sn}, \mathrm{v}, \pi$ — Consume the value-v coin w/ serial number sn.

Here is a ZK proof π that I know secret $(\mathrm{cm}, \mathrm{k}, \mathrm{r}, \mathrm{s}, \rho, \mathrm{apk}, \mathrm{ask})$ s.t.

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



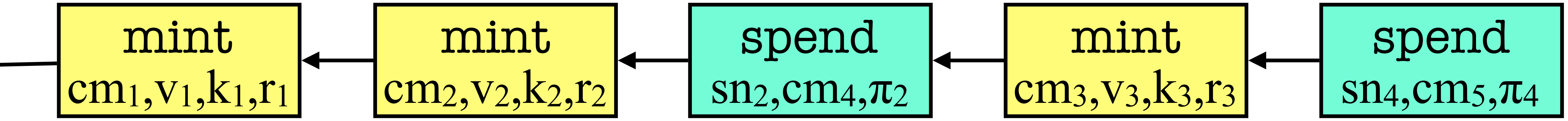
Good:

cannot double spend
others can't spend my coins
spend and mint unlinkable
variable denomination

Bad:

still only hides sender
…

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.

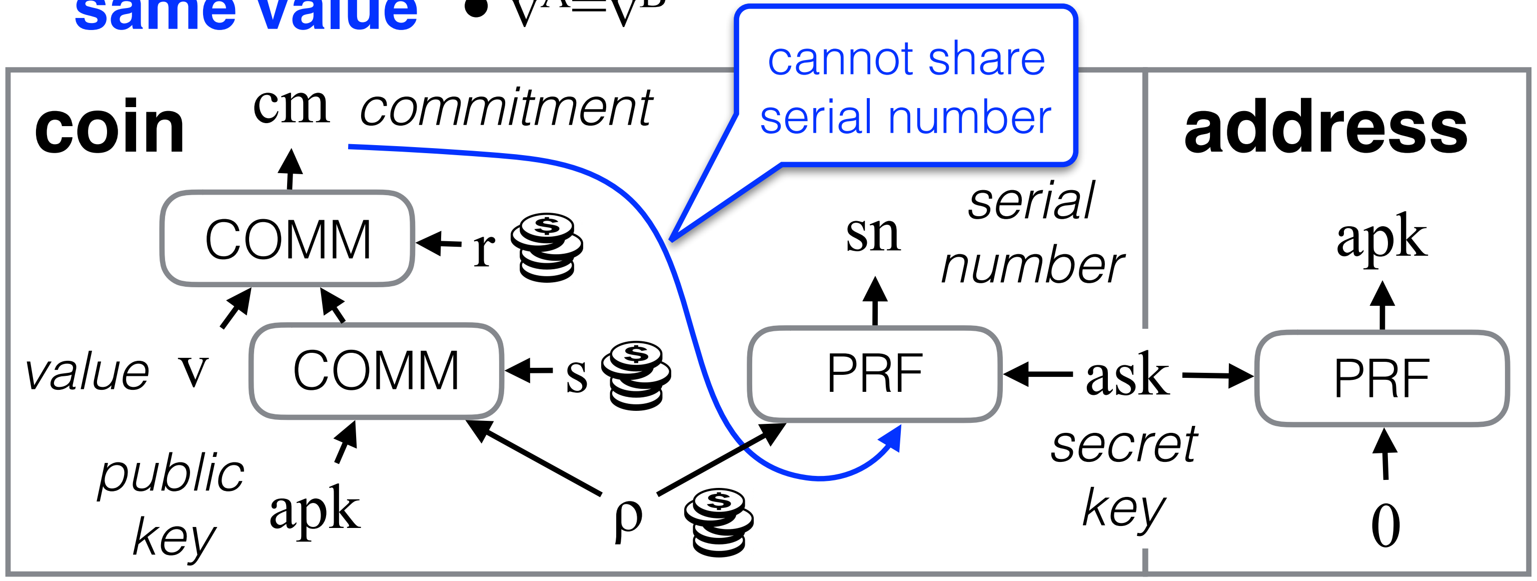
spend sn^A,cm^B,π — Consume coin w/ serial number sn^A & create coin w/ commitment cm^B.

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

$(cm^B,v^B,k^B,r^B,s^B,\rho^B,apk^B)$

send out-of-band or via blockchain

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

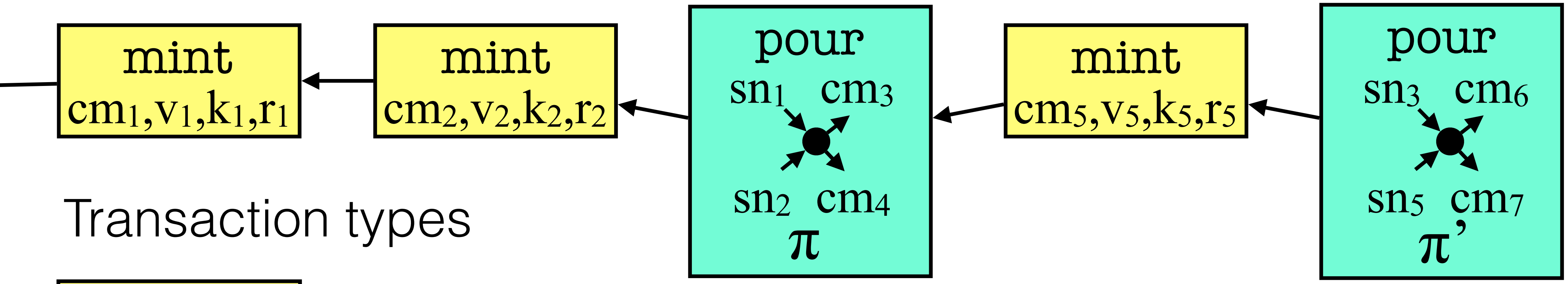


Good:

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

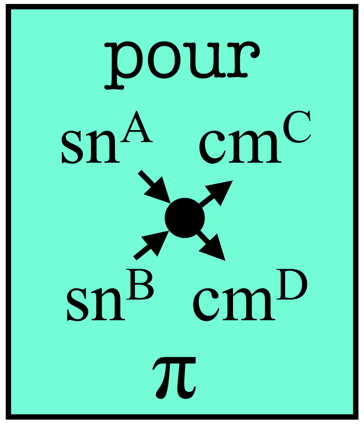


view of blockchain

Transaction types

mint
cm, v, k, r

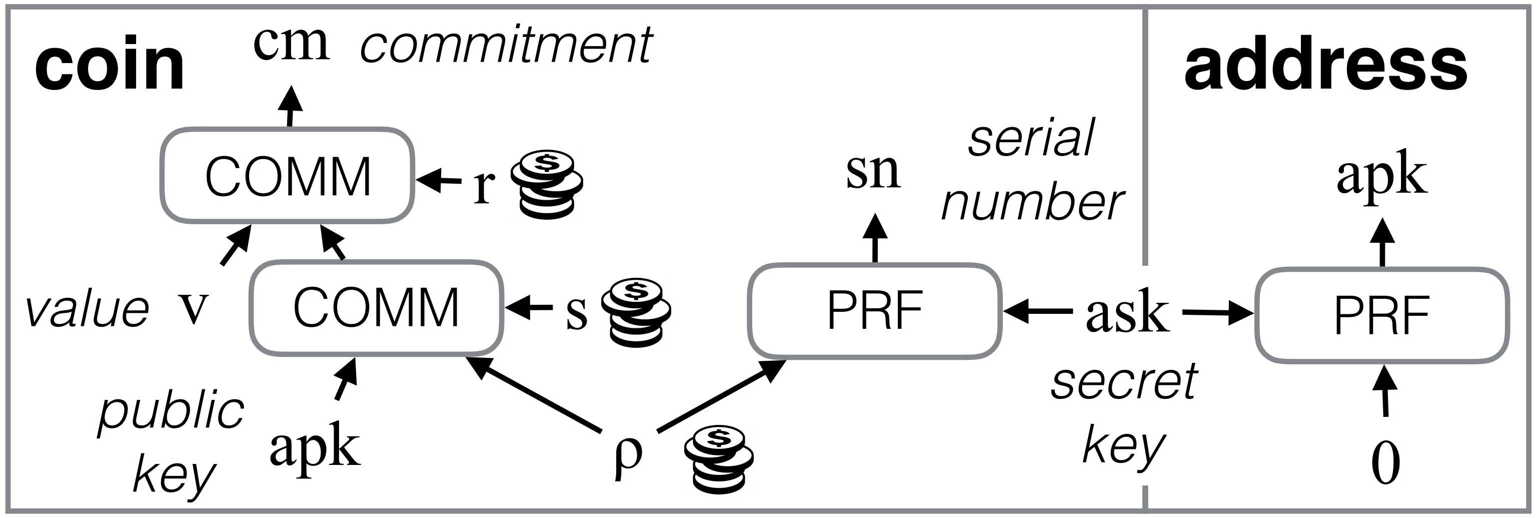
Consume v BTC to create a value-v coin w/ commitment cm.



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

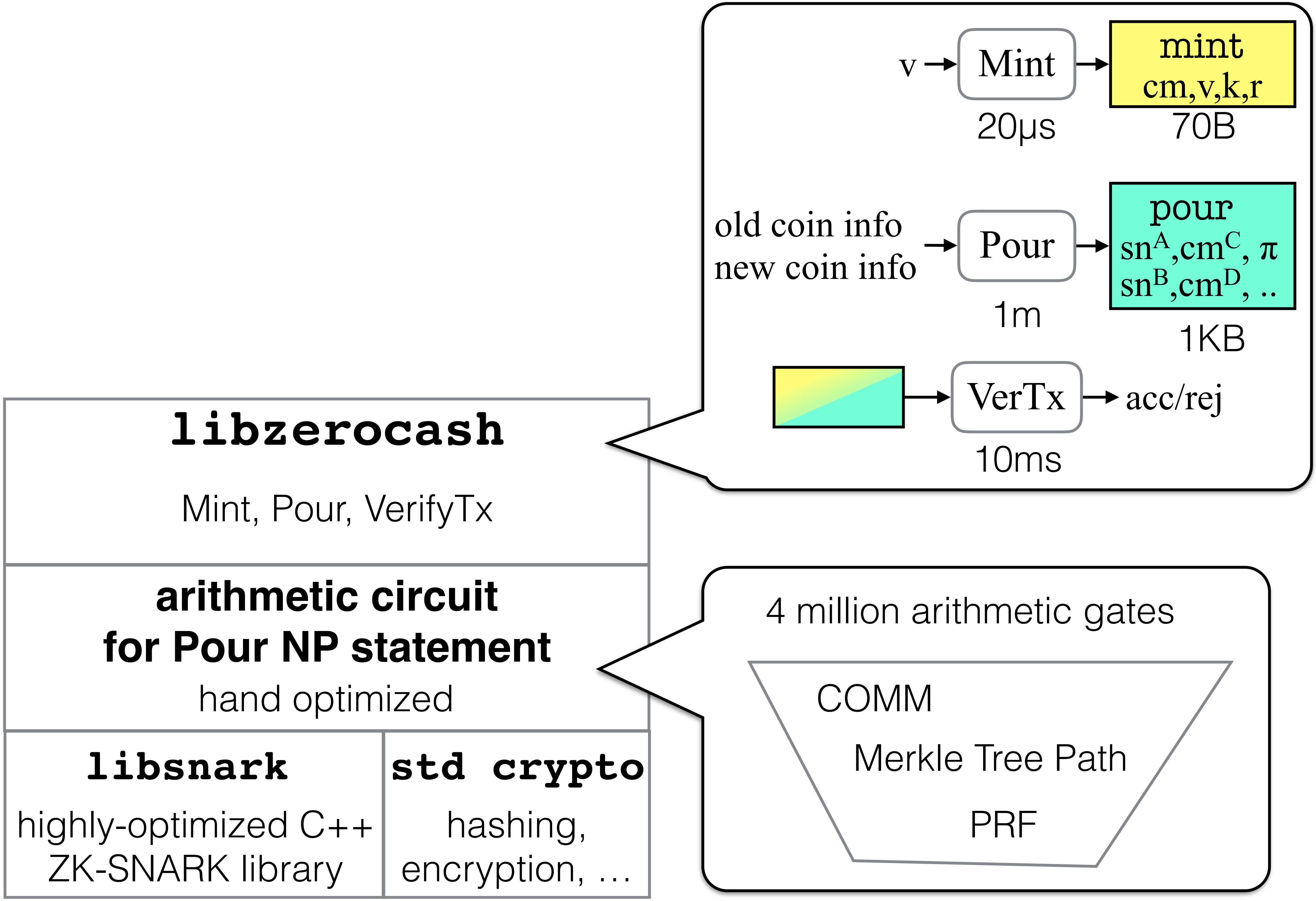
✓ split coins

✓ making change

✓ pay transaction fees

Deployment

Proof-of-concept implementation



v → Mint → mint cm,v,k,r

20μs

70B

old coin info
new coin info → Pour → pour sn^A, cm^C, π sn^B, cm^D, ..

1m

1KB

→ VerTx → acc/rej

10ms

libzerocash

Mint, Pour, VerifyTx

arithmetic circuit for Pour NP statement

hand optimized

libsnark

highly-optimized C++ ZK-SNARK library

std crypto

hashing, encryption, …

4 million arithmetic gates

COMM

Merkle Tree Path

PRF

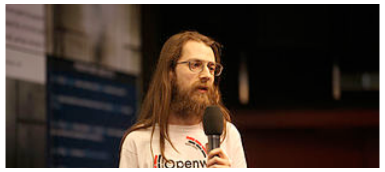
Academic Practical → Real-World Practical

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

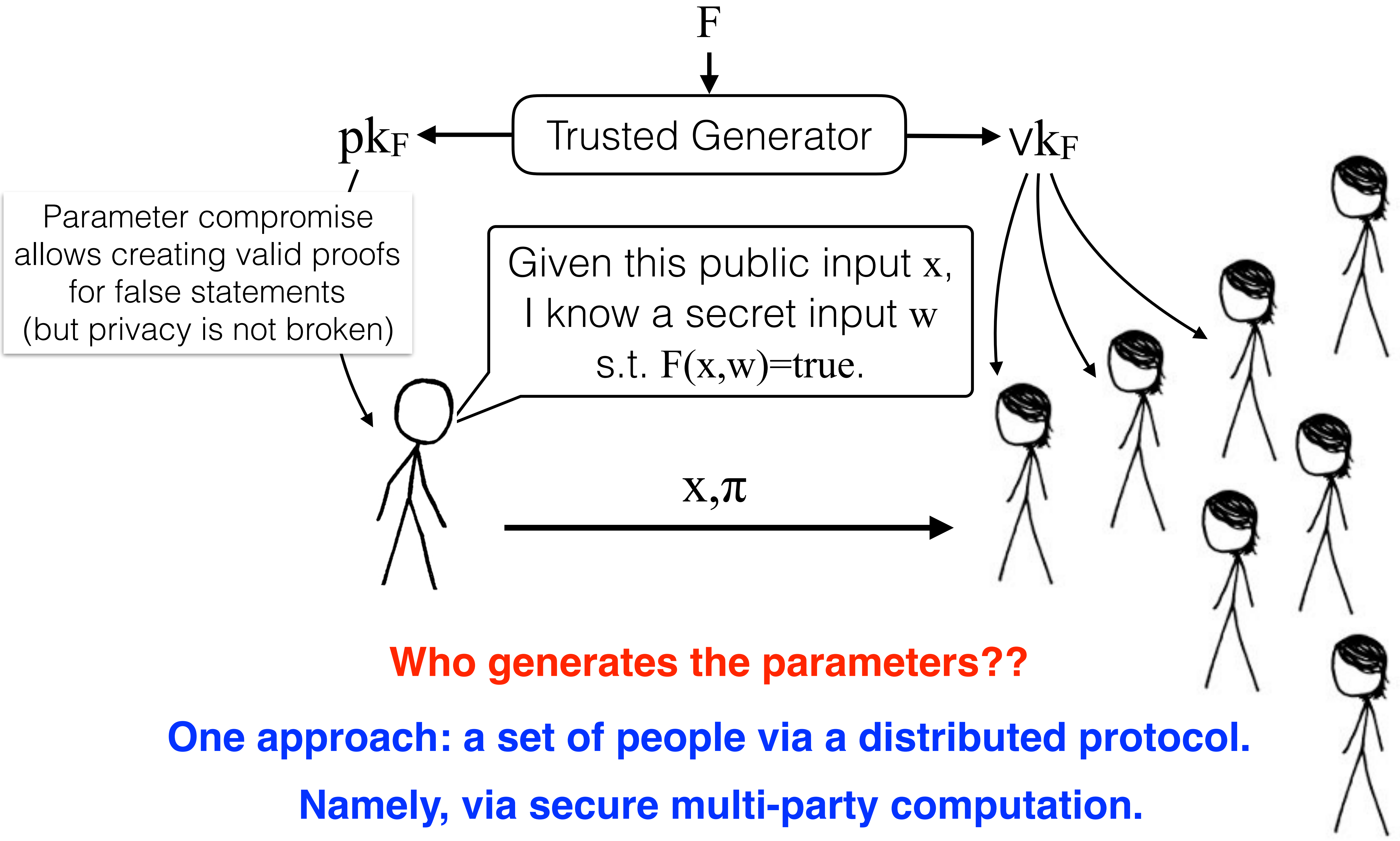


Solar Designer
(Alexander Peslyak)

- 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.



F

Trusted Generator

pk_F

vk_F

Parameter compromise allows creating valid proofs for false statements (but privacy is not broken)

Given this public input x, I know a secret input w s.t. $F(x,w)$=true.

x,π

Who generates the parameters??

One approach: a set of people via a distributed protocol.

Namely, via secure multi-party computation.

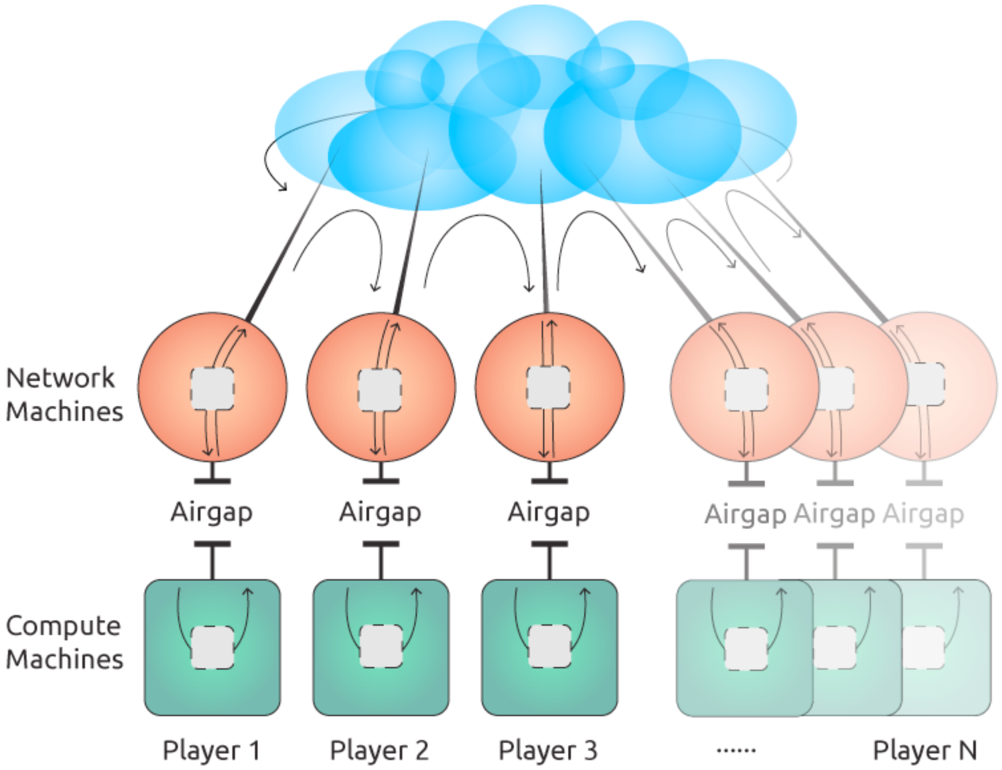
MPC Ceremony

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



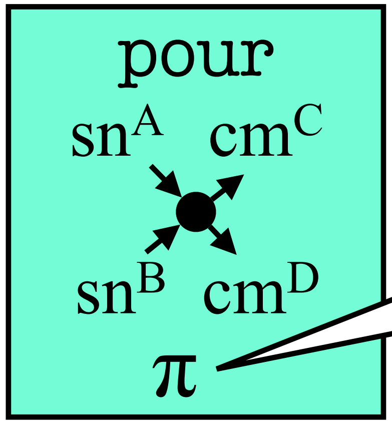
n=6 geographically distributed participants (including one security company, and a mobile station)

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

video documentation from most participants including destruction of compute nodes

Frontiers

Beyond Privacy & Fungibility



I'm consuming my **unspent** coins in order to create new coins in a way that **value is preserved**.
I'm not revealing the value, sender, or receiver.

& the receiver was a 501(c) organization but I am not revealing which one

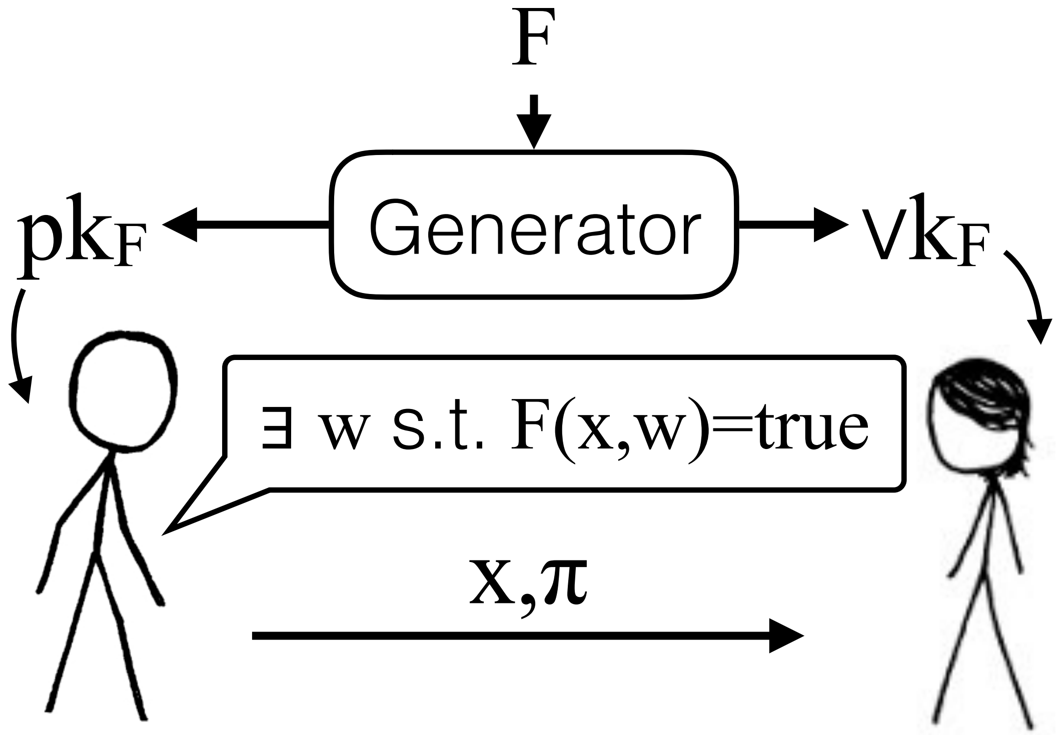
& the value transfered lies in [10,20]

Exciting research direction:

Which policies are desirable (and feasible!) to balance privacy/fungibility and oversight/accountability?

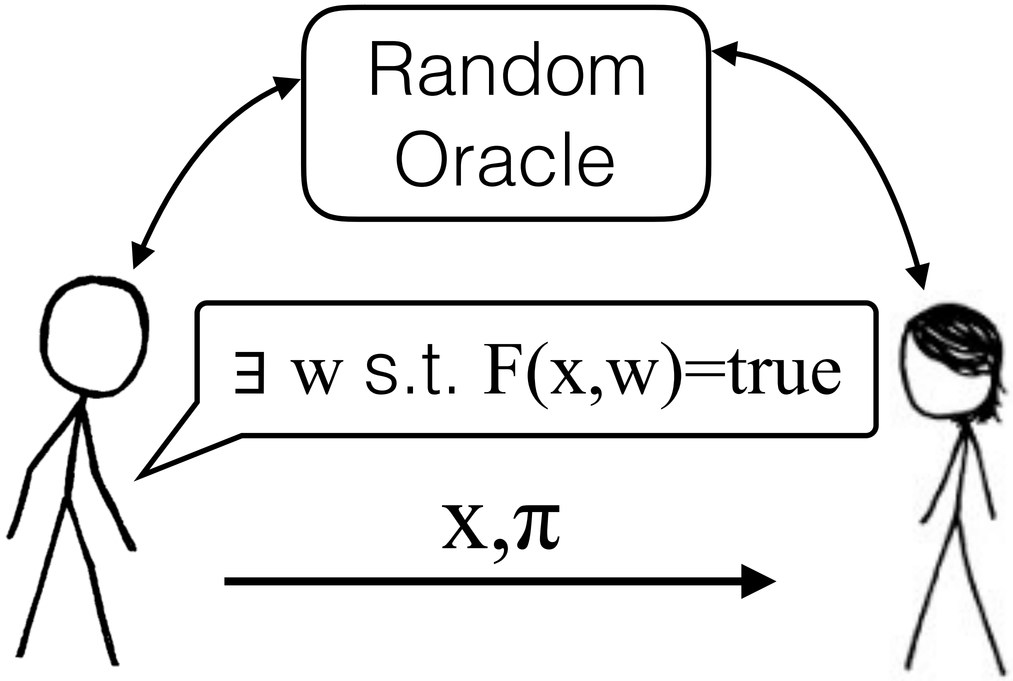
ZK-SNARKs with Public Setup

Current practical ZK-SNARKs



There are other constructions…

BUT



Main obstacle is concrete efficiency.

Based on probabilistic checking techniques,

and more research is needed to “scale down” to practice.

Lots of fun problems in complexity theory / property testing.

Thanks!