Zerocash

Decentralized Anonymous Payments from Bitcoin

Eli Ben-Sasson (Technion)

Alessandro Chiesa (MIT)

Christina Garman (JHU)

Matthew Green (JHU)

Ian Miers (JHU)

Eran Tromer (Tel Aviv University)

Madars Virza (MIT)

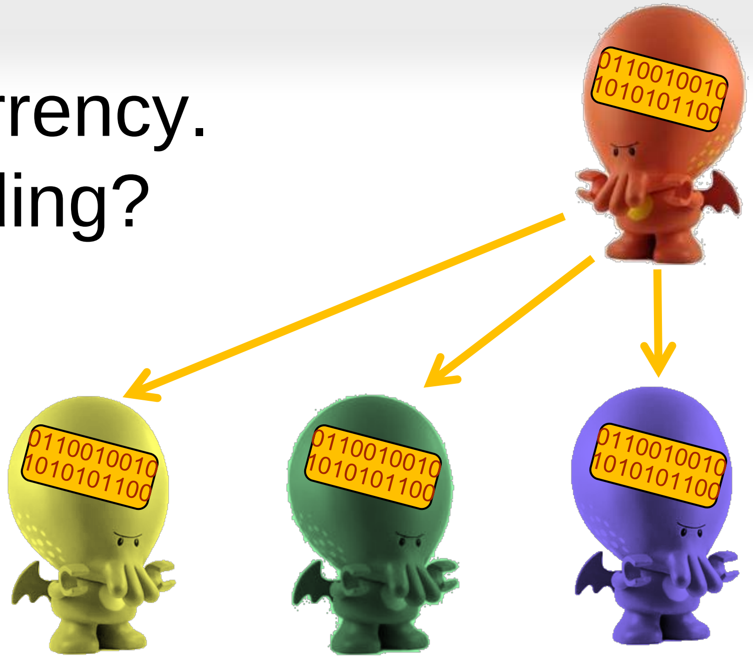
`zerocash-project.org`

IEEE Symposium on Security and Privacy 2014 20 May 2014

Bitcoin's privacy problem

Bitcoin: decentralized <u>digital</u> currency.

What's to prevent double-spending?

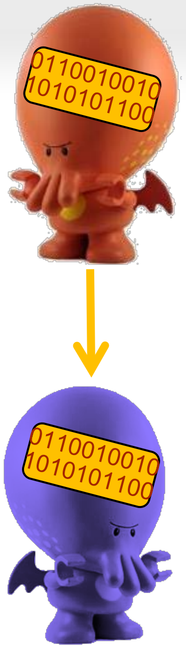


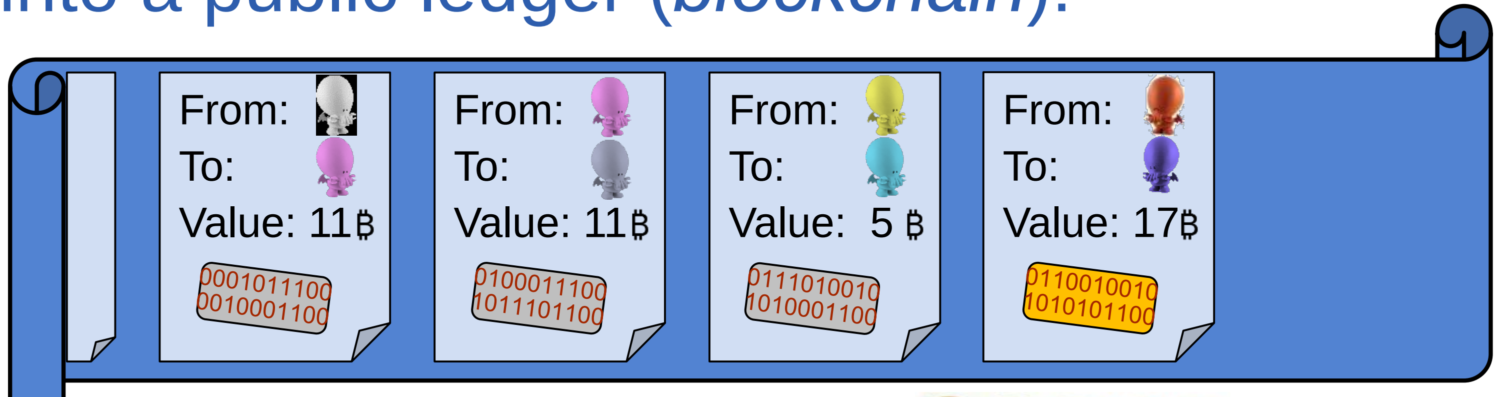
Bitcoin’s privacy problem

Bitcoin: decentralized digital currency.

What’s to prevent double-spending?

Solution: broadcast every transaction into a public ledger (*blockchain*):





From:
To:
Value: 11฿

From:
To:
Value: 11฿

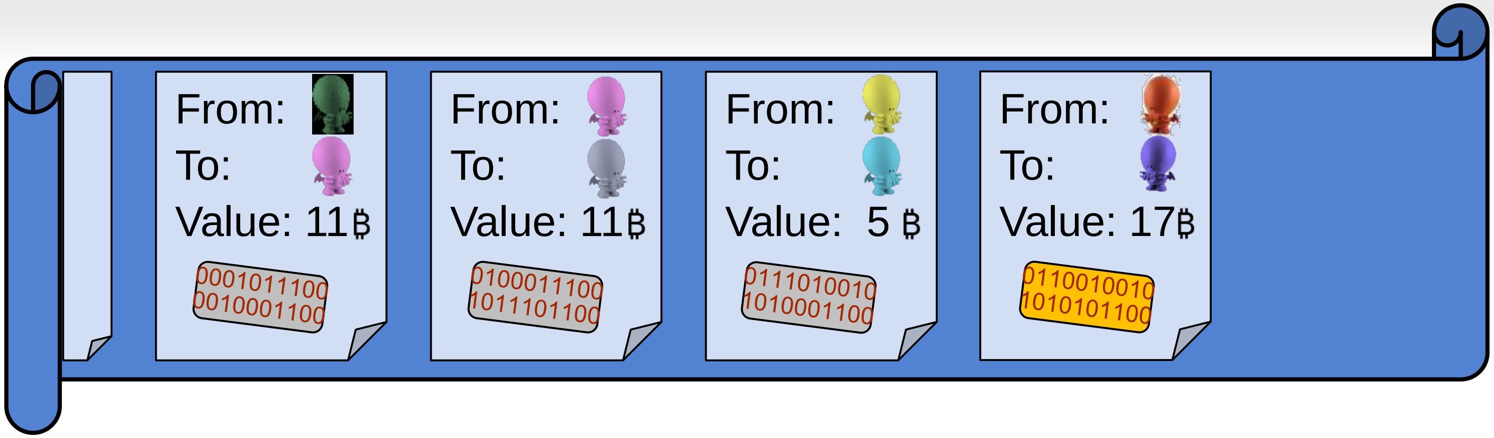
From:
To:
Value: 5 ฿

From:
To:
Value: 17฿

The cost: privacy.

- Consumer purchases (timing, amounts, merchant) seen by friends, neighbors, and co-workers.
- Account balance revealed in every transaction.
- Merchant’s cash flow exposed to competitors.

Bitcoin’s privacy problem (cont.)



- Pseudonymous, but:
  - Most users use a single or few addresses
  - Transaction graph can be analyzed.
    [Reid Martin 11] [Barber Boyen Shi Uzun 12] [Ron Shamir 12] [Meiklejohn PJLMVS 13]
- Also: threat to the currency’s fungibility.
- Centralized: reveal to the bank.
- Decentralized: reveal to everyone?!

Past attempts at Bitcoin anonymity

- Trusted mix (but: operator can trace/steal)
- Zerocoin: decentralized mix service for Bitcoin

  [Miers Garman Green Rubin 13]

  Limitations:

  – Performance: 45 kB/spend (to be broadcast, verified, and stored in blockchain), take ~0.5 s to verify. (for 128-bit security)

  – Single denomination (undivisible) ⇒ reveals amount

  – Reveal payment destinations; no direct transfer

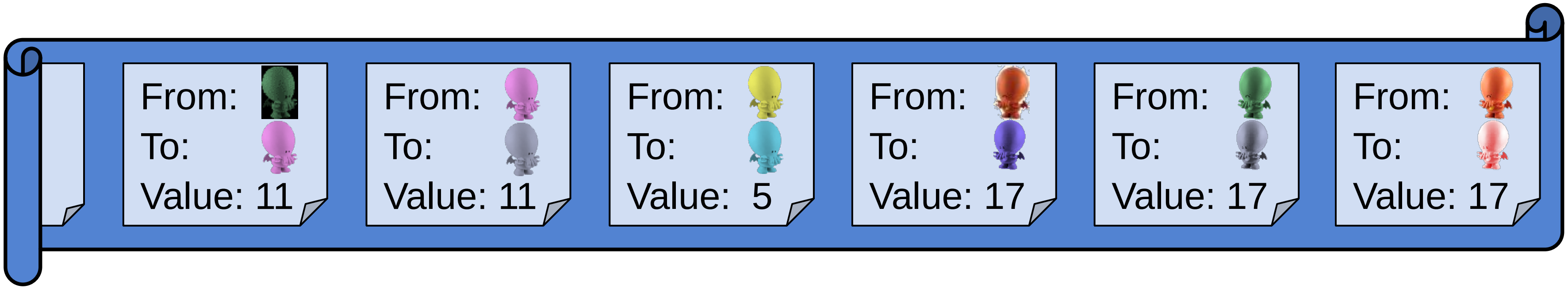
  – Requires explicit “laundry” process.
- Pinocchio Coin: variant using “Pinocchio” ZK proofs: 344 B/spend [Danezis Fournet Kohlweiss Parno 13]

  – Scalability problem: spend time grows linearly with #coins

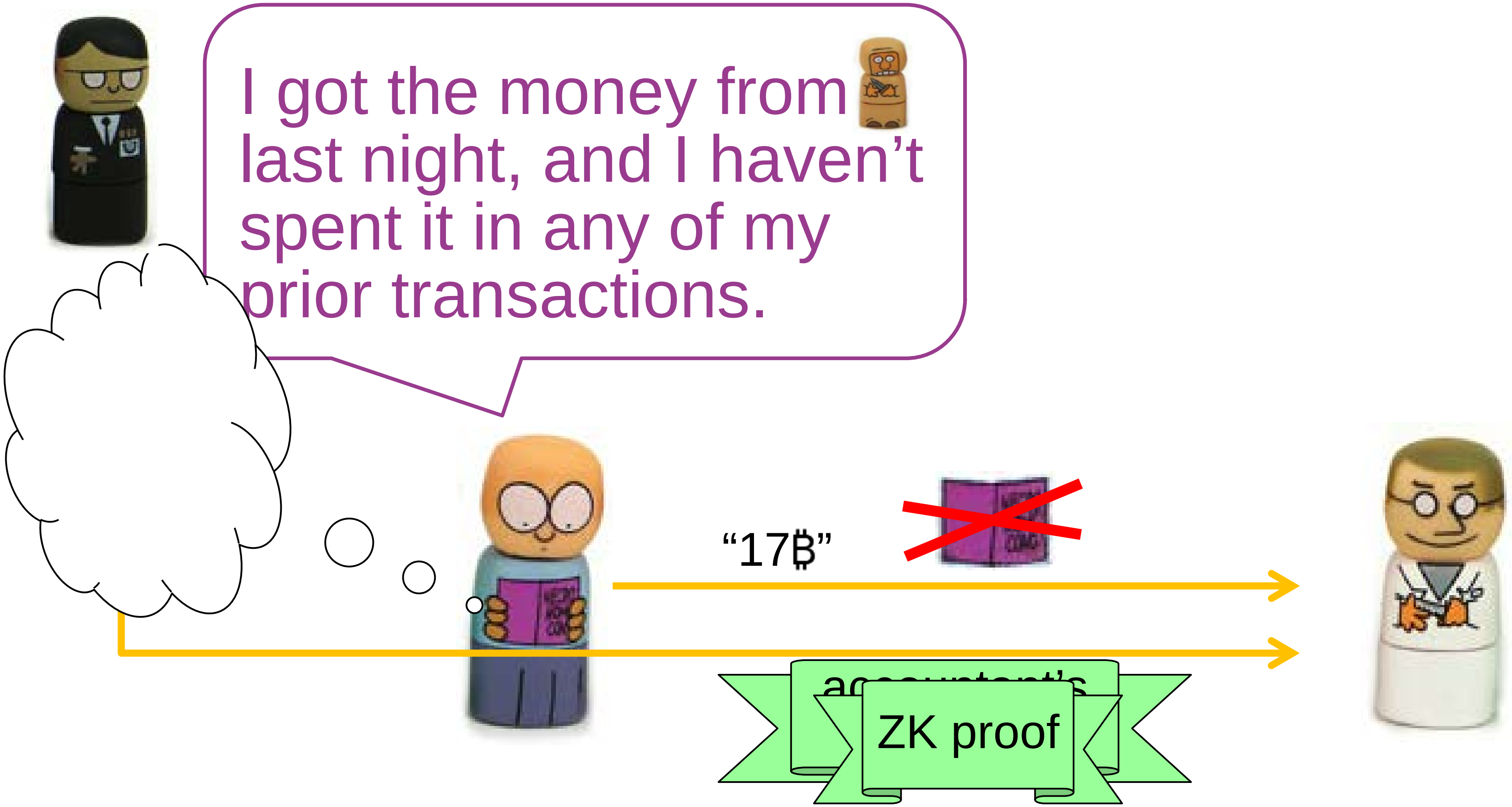
  – Still single denomination, reveals amount, reveals destination, explicit.
- CoinJoin and various other mixing/pooling solutions
- Goal: fully preserves privacy, efficient, transparent, always on.

Zerocash: divisible anonymous payments

- Zerocash is a new privacy-preserving protocol for digital currency designed to sit on top of *Bitcoin* (or similar ledger-based currencies).
- Zerocash enables users to pay one another directly via payment transactions of variable denomination that reveal neither the origin, destination, or amount.



Zerocash: in proofs we trust



I got the money from last night, and I haven't spent it in any of my prior transactions.

Intuition: "virtual accountant" using cryptographic proofs.

More about Zerocash

- Efficiency:
  - 288 proof bytes/spend at 128-bit security level,
  - <6 ms to verify a proof
  - <1 min to create for 2^{64} coins; asymptotically: log(#coins)
  - 896MB “system parameters” (fixed throughout system lifetime).
- Trust in initial generation of system parameters (once).
- Crypto assumptions:
  - Pairing-based elliptic-curve crypto
  - Less common: Knowledge of Exponent [Boneh Boyen 04] [Gennaro 04] [Groth 10]
  - Properties of SHA256, encryption and signature schemes

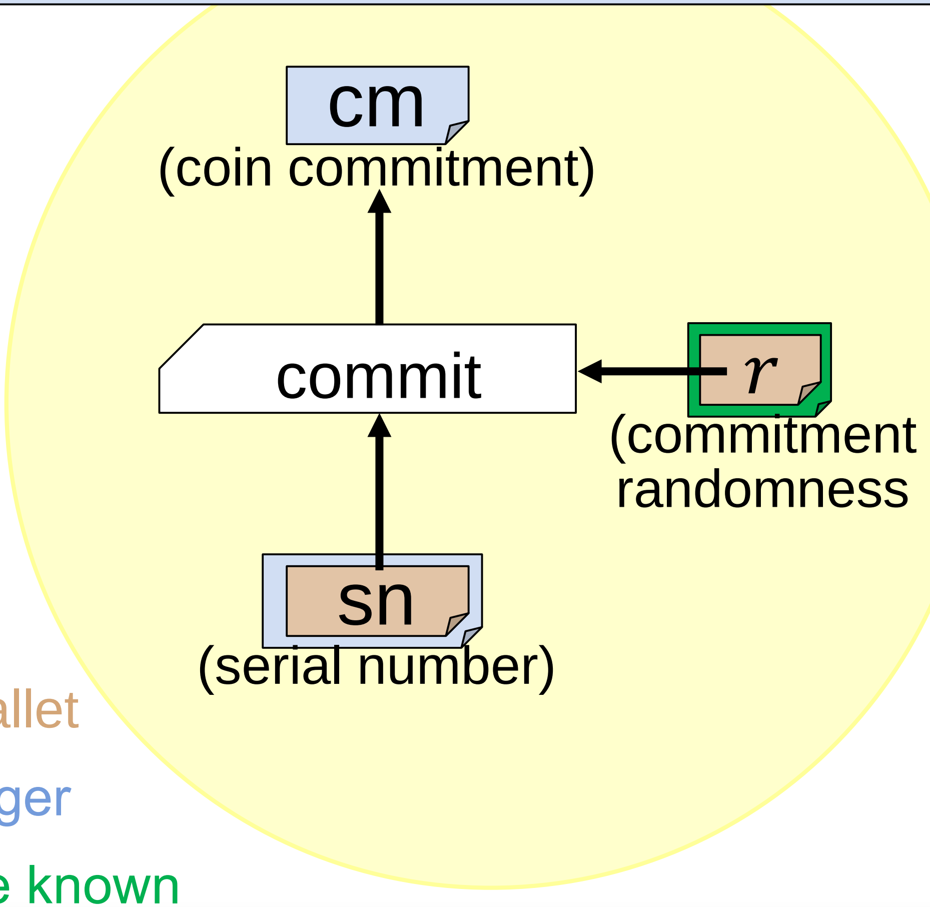
The Zerocash scheme

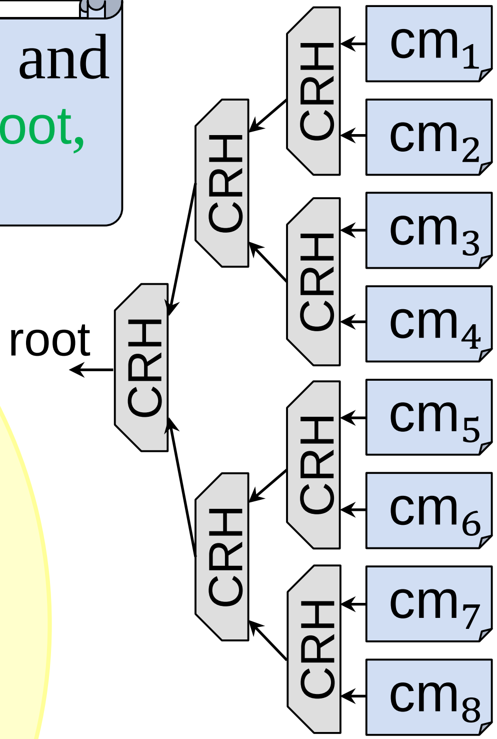
Basic anonymous e-cash

[Sander Ta-Shma 1999]

Minting: I hereby spend 1 BTC to create cm

Spending: I'm using up a coin with (unique) sn, and I know r, and a cm in the tree with root, that match sn.





Legend:

🟫 In private wallet

🟦 In public ledger

🟩 Proved to be known

Basic anonymous e-cash – requisite proofs

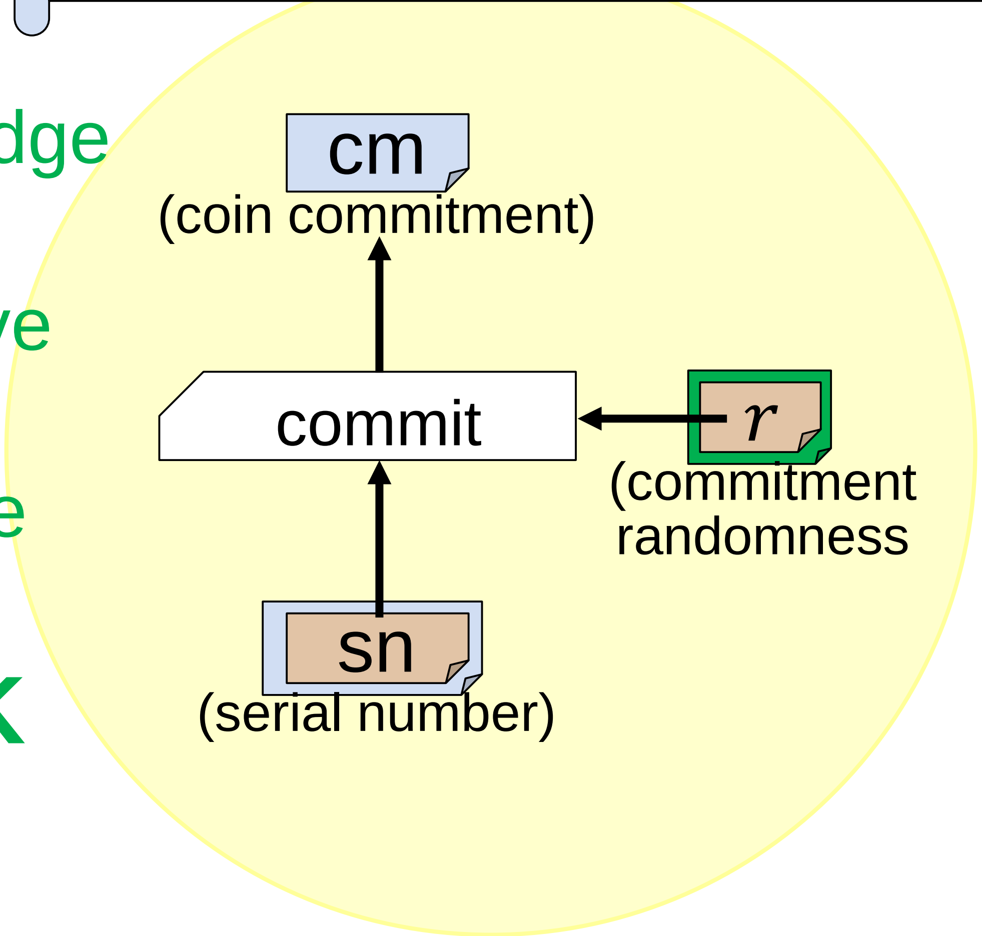
Spending:

I'm using up a coin with (unique) sn,
and I know a cm in the tree, and r,
that match sn.

Requires:

zero **k**nowledge
succinct
noninteractive
argument
proof
of **k**nowledge

zkSNARK



zkSNARK constructions for any *NP statement*

zero **k**nowledge
succinct
noninteractive
argument
of **k**nowledge

zkSNARK

Without trusted setup:

– Theory [BFLS 91] [Kilian 92] [Micali 94] […PCP…] [Ben-Sasson Chiesa Genkin Tromer 13]

With trusted setup:

– Theory [Groth 10] [Lipmaa 12] [Gennaro Gentry Parno Raykova 13] [Bitansky Chiesa Ishai Ostrovsky Paneth 13]

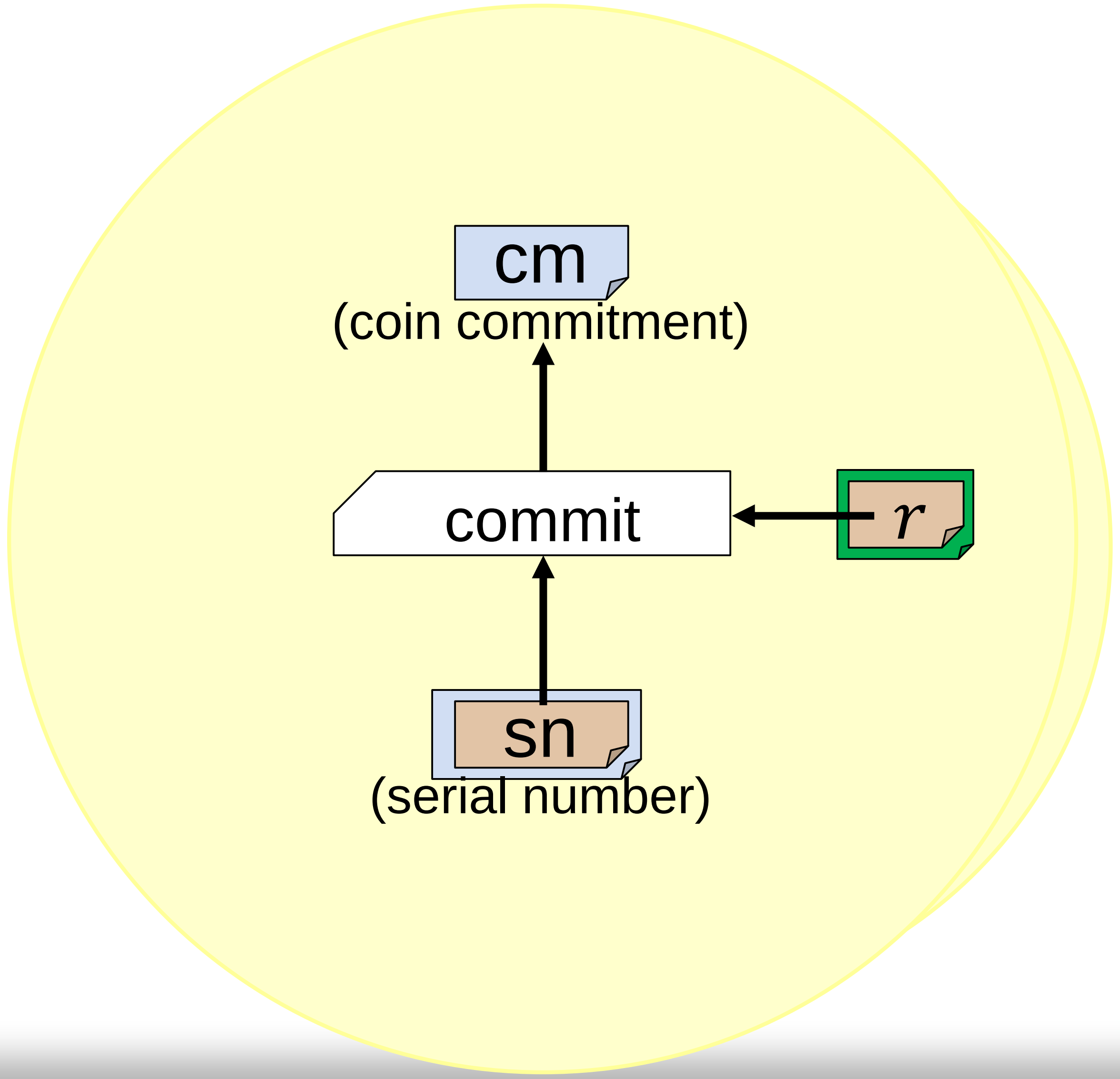
– Implementations [Parno Gentry Howell Raykova 13] [Ben-Sasson Chiesa Genkin Tromer Virza 13]

SCIPR LAB [Ben-Sasson Chiesa Tromer Virza 14]

Underlying zkSNARK used in Zerocash

zkSNARK

with great power comes great functionality

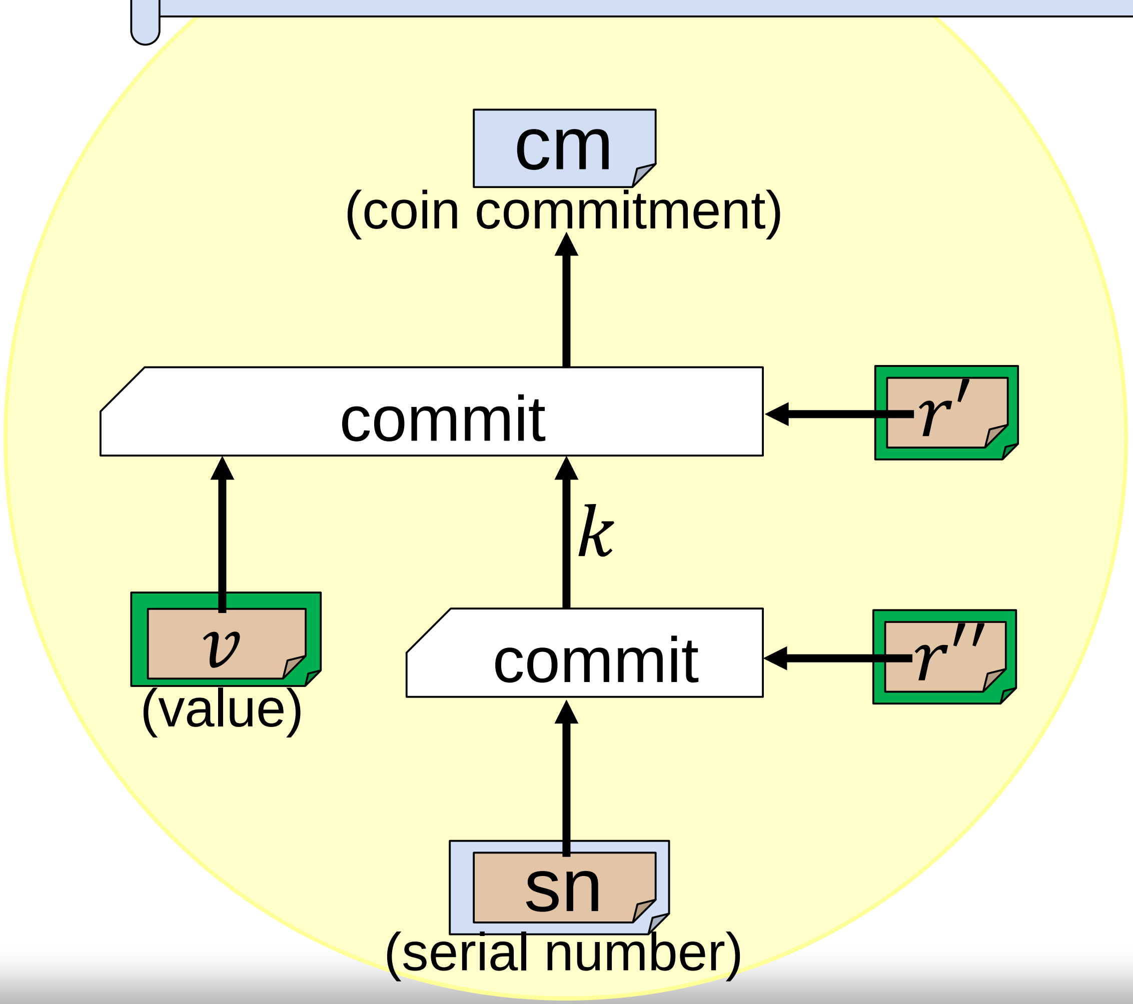


Adding variable denomination

Minting: I hereby spend v BTC to create cm, and here is k, r' to prove consistency.

Spending: I'm using up a coin with value v (unique) sn, and I know r', r'' that are consistent with cm.

zkSNARK



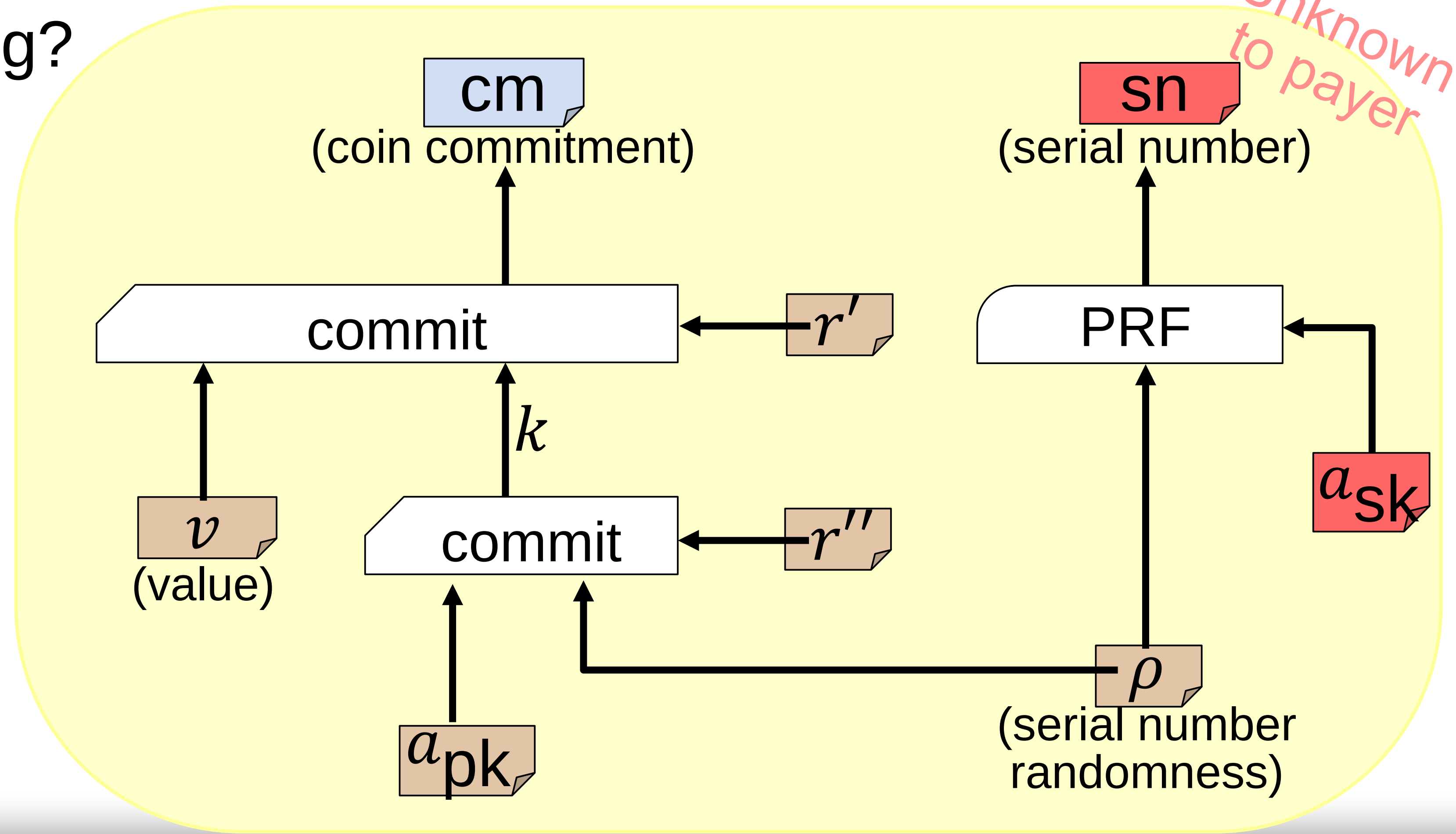
Adding direct anonymous payments

CreateAddress: payee creates $a_{\mathsf{pk}}, a_{\mathsf{sk}}$

Minting, spending analogous to above.

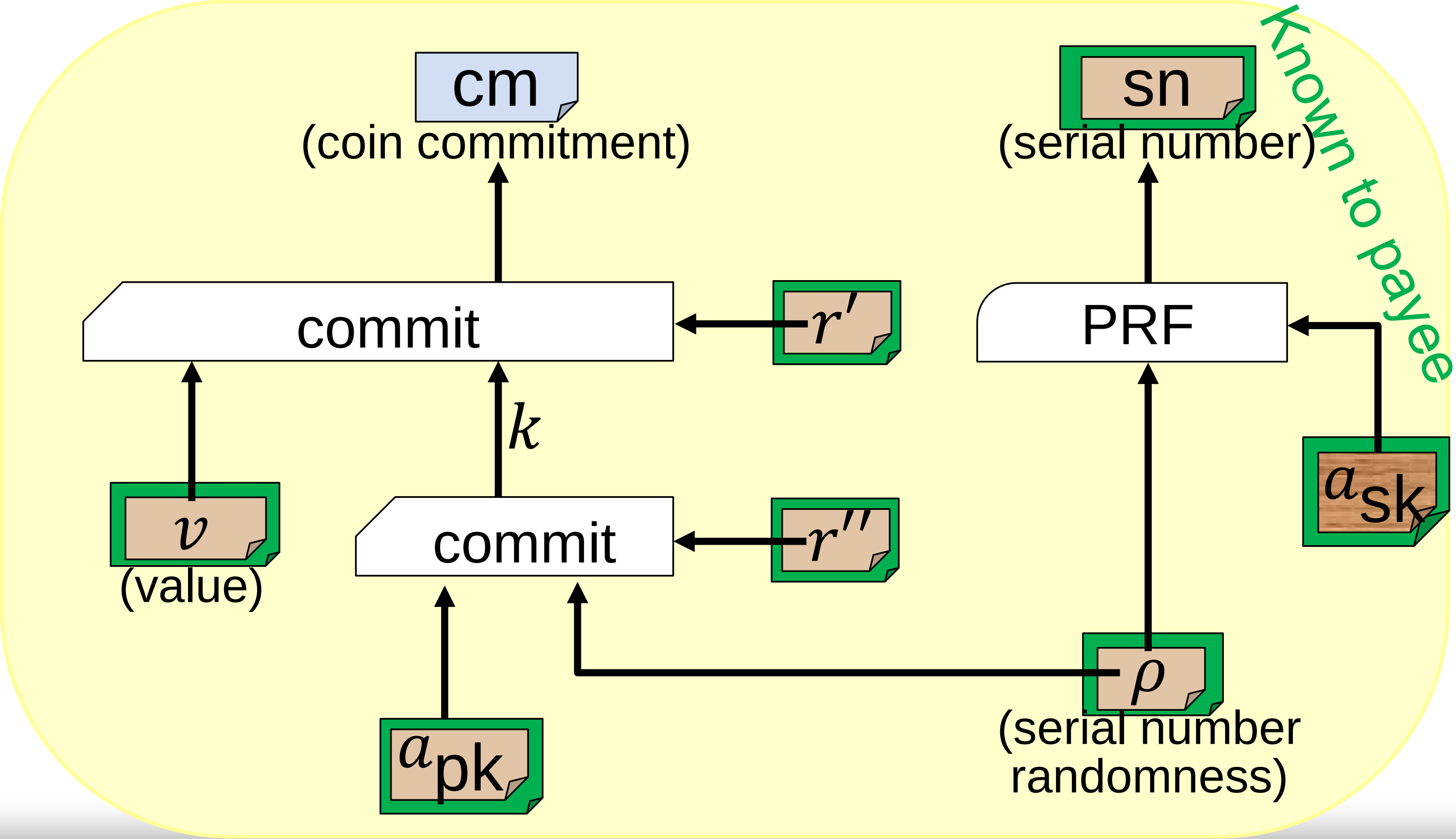
Sending?

I'm using up a coin with value v (unique) sn, and I know $r', r'', \rho, a_{\mathsf{pk}}$ that are consistent with cm.



Sending direct anonymous payments

1. Create coin using a_{pk} of payee.
2. Send coin secrets (v, ρ, r', r'') to payee
   out of band, or encrypted to payee's public key.



Pouring Zerocash coins

Simplified

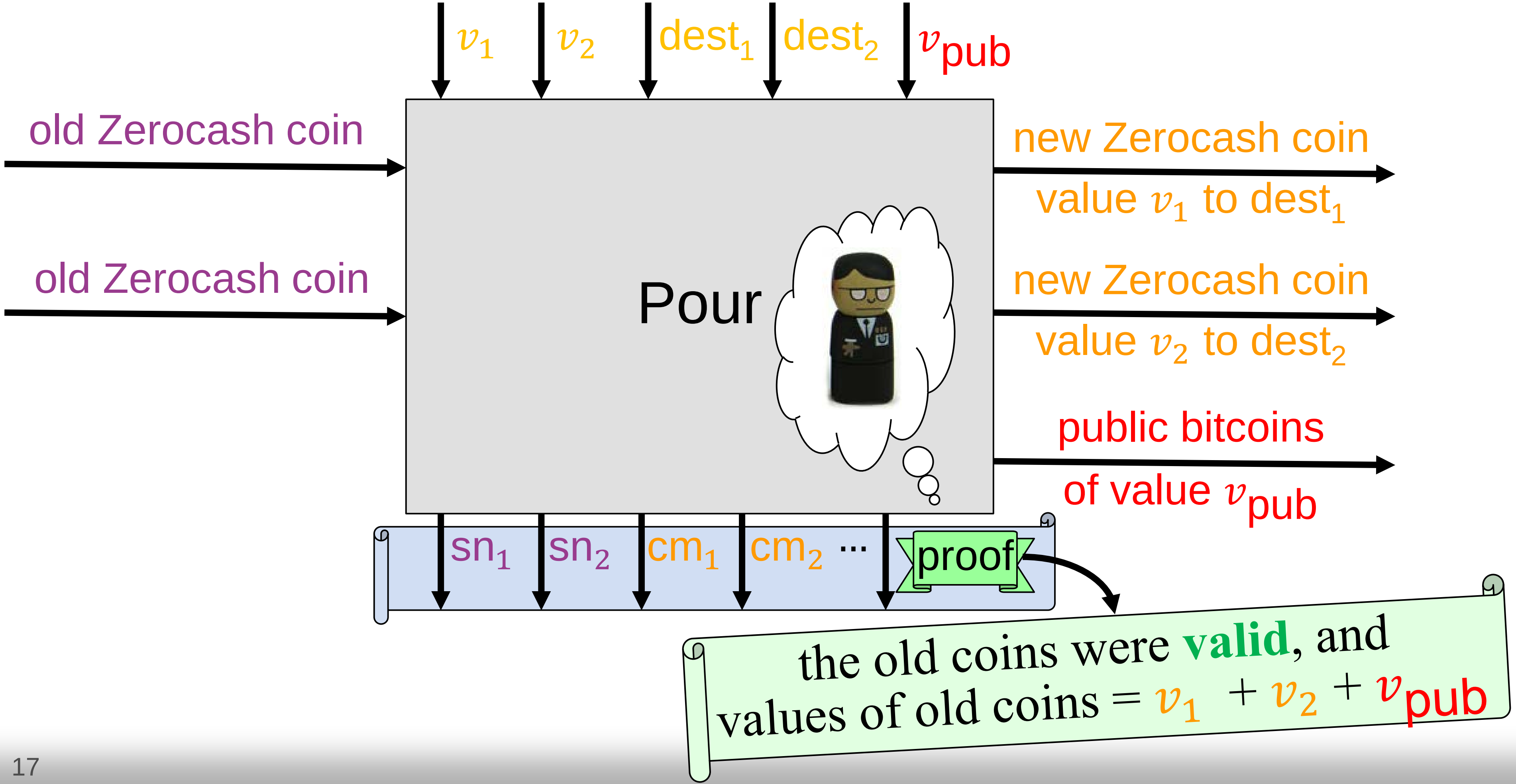
Single transaction type capturing:

Sending payments

Making change

Exchanging into bitcoins

Transaction fees



v_1 v_2 dest_1 dest_2 v_{pub}

old Zerocash coin

old Zerocash coin

Pour

new Zerocash coin value v_1 to dest_1

new Zerocash coin value v_2 to dest_2

public bitcoins of value v_{pub}

sn_1 sn_2 cm_1 cm_2 ··· proof

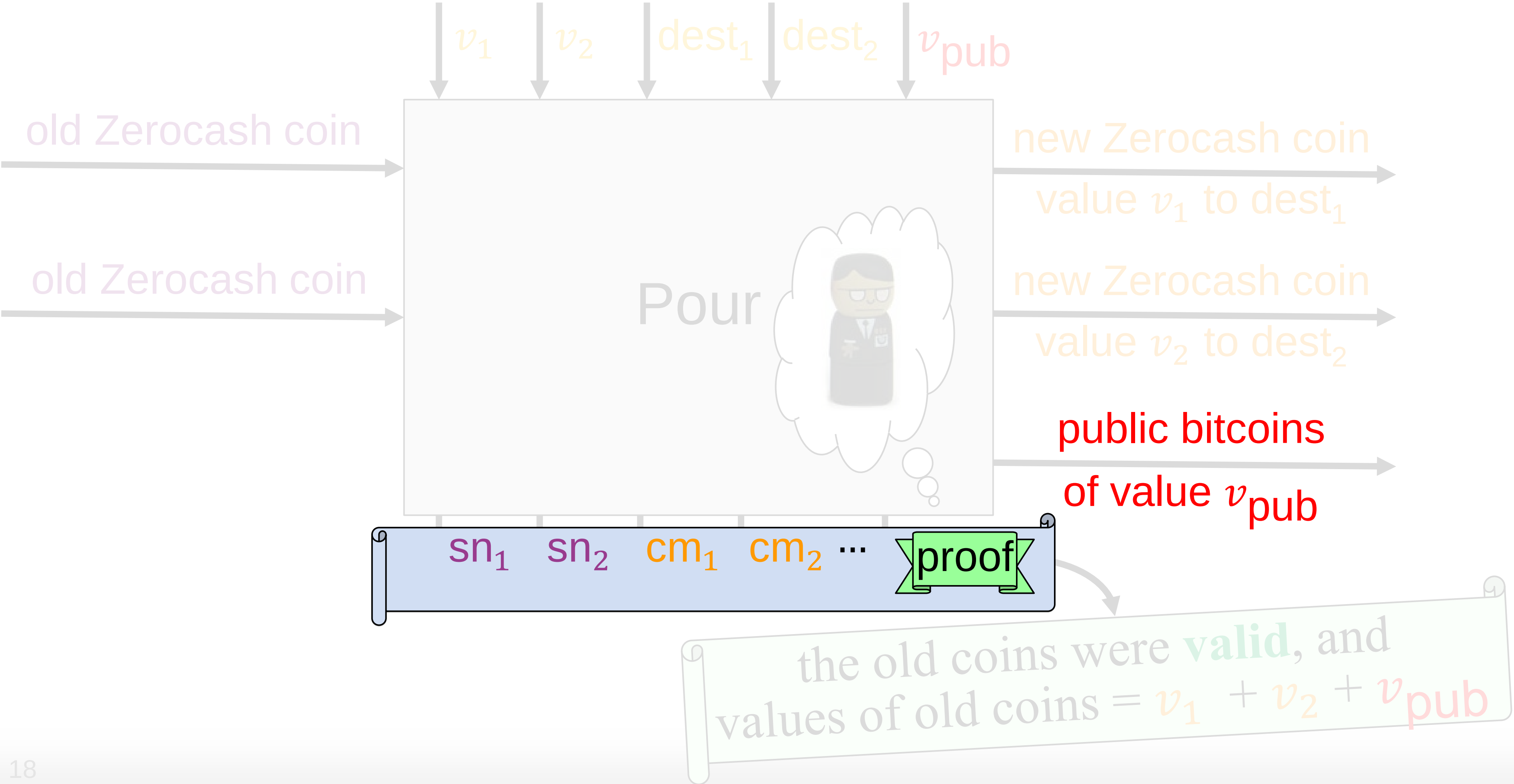
the old coins were **valid**, and values of old coins = $v_1 + v_2 + v_{\text{pub}}$

Pouring Zerocash coins

Simplified

Single transaction type capturing:

- Sending payments
- Making change
- Exchanging into bitcoins
- Transaction fees



v_1
v_2
dest_1
dest_2
v_{pub}
old Zerocash coin
old Zerocash coin
Pour
new Zerocash coin
value v_1 to dest_1
new Zerocash coin
value v_2 to dest_2
public bitcoins
of value v_{pub}
sn_1 sn_2 cm_1 cm_2 ···
proof
the old coins were valid, and
values of old coins = $v_1 + v_2 + v_{\text{pub}}$

Example of a Zerocash Pour transaction

root	1c4ac4a110e863deeca050dc5e5153f2b7010af9
sn_1	a365e7006565f14342df9096b46cc7f1d2b9949367180fdd8de4090eee30bfdc
sn_2	6937031dce13facdebe79e8e2712ffad2e980c911e4cec8ca9b25fc88df73b52
cm_1	a4d015440f9cfae0c3ca3a38cf04058262d74b60cb14ecd6063e047694580103
cm_2	2ca1f833b63ac827ba6ae69b53edc855e66e2c2d0a24f8ed5b04fa50d42dc772
v_pub	0000000000000042
pubkeyHash info	8f9a43f0fe28bef052ec209724bb0e502ffb5427
SigPK	2dd489d97daa8ceb006cb6049e1699b16a6d108d43
Sig	f1d2d2f924e986ac86fdf7b36c94bcdf32beec15a38359c82f32dbb3342cb4bedcb78ce116bac69e
MAC_1	b8a5917eca1587a970bc9e3ec5e395240ceb1ef700276ec0fa92d1835cb7f629
MAC_2	ade6218b3a17d609936ec6894b7b2bb446f12698d4bcafa85fcbf39fb546603a
ciphertext_1	048070fe125bdaf93ae6a7c08b65adbb2a438468d7243c74e80abc5b74dfe3524a987a2e3ed075d54ae7a53866973eaa5070c4e0895 4ff5d80caae214ce572f42dc6676f0e59d5b1ed68ad33b0c73cf9eac671d8f0126d86b667b319d255d7002d0a02d82efc47fd8fd648 057fa823a25dd3f52e86ed65ce229db56816e646967baf4d2303af7fe09d24b8e30277336cb7d8c81d3c786f1547fe0d00c029b63bd 9272aad87b3f1a2b667fa575e
ciphertext_2	0493110814319b0b5cabb9a9225062354987c8b8f604d96985ca52c71a77055b4979a50099cefc5a359bdf0411983388fa5de840a0d 64816f1d9f38641d217986af98176f420caf19a2dc18c79abcf14b9d78624e80ac272063e6b6f78bc42c6ee01edfbcddbeb60eba586 eaecd6cb017069c8be2ebe8ae8a2fa5e0f6780a4e2466d72bc3243e873820b2d2e4b954e9216b566c140de79351abf47254d122a35f 17f840156bd7b1feb942729dc
zkSNARKproof	a4c3cad6e02eec51dc8a37ebc51885cf86c5da04bb1c1c0bf3ed97b778277fb8adceb240c40a0cc3f2854ce3df1eafdcefccc532bc5afaefefe9d3975726f2ca829228 6ca8dd4f8da21b3f98c61fac2a13f0b82544855b1c4ce7a0c9e57592ee1d233d43a2e76b9bdeb5a365947896f117002b095f7058bdf611e20b6c2087618c58208e3 658cfcc00846413f8f355139d0180ac11182095cdee6d9432287699e76ed7832a5fc5dc30874ff0982d9658b8e7c51523e0fa1a5b649e3df2c9ff58dc05dac7563741 298025f806dfbe9cfe5c8c40d1bf4e87dacb11467b9e6154fb9623d3fba9e7c8ad17f08b17992715dfd431c9451e0b59d7dc506dad84aef98475d4be530eb501925 dfd22981a2970a3799523b99a98e50d00eaab5306c10be5

~1KB total. Less without direct payments and public outputs.

Decentralized Anonymous Payment (DAP) system

Algorithms:

`Setup` `CreateAddress`

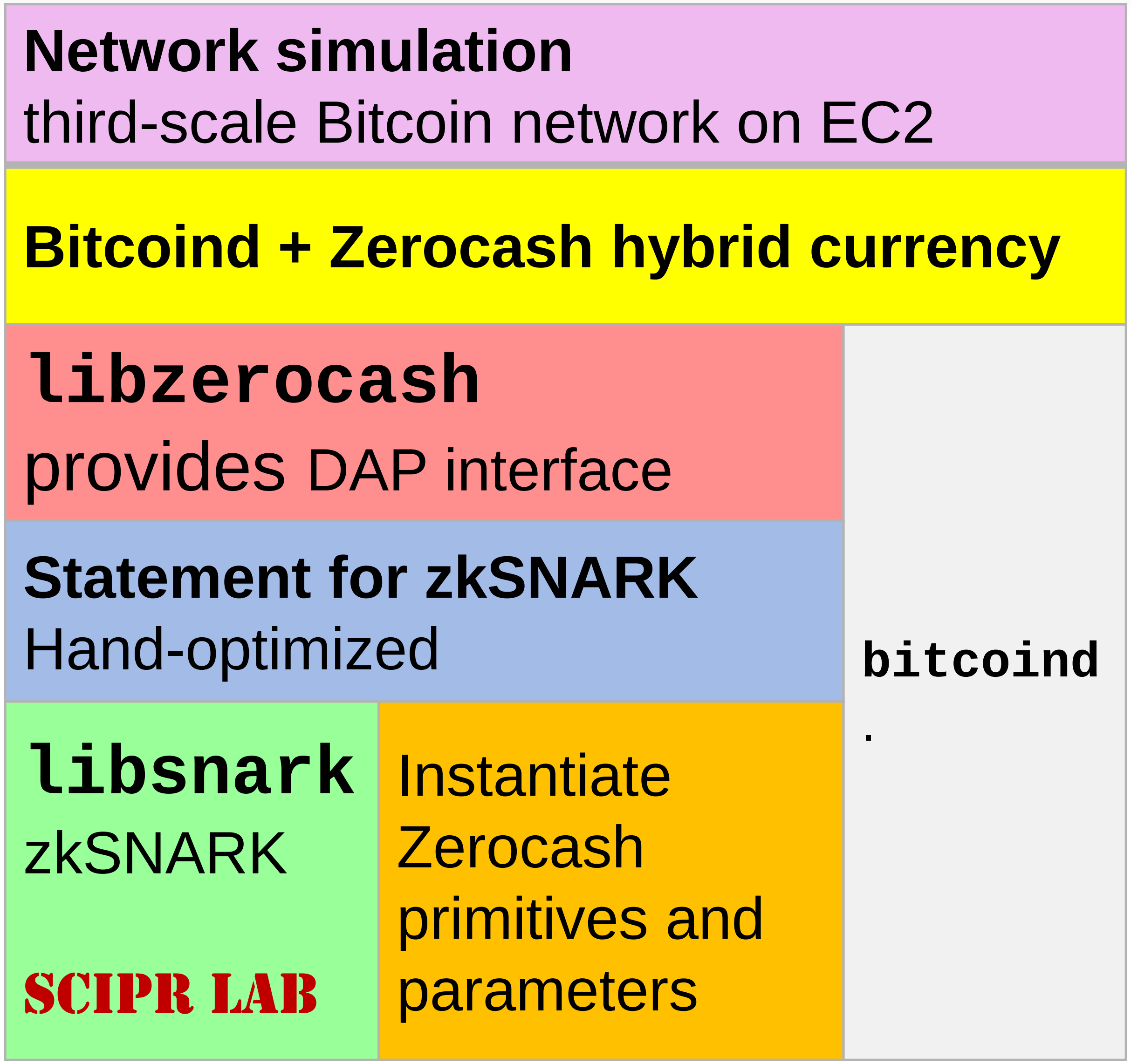
`Mint` `Pour`

`VerifyTransaction` `Receive`

Security:

1. Ledger indistinguishability
   Nothing revealed beside public information, even by chosen-transaction adversary.
2. Balance
   Can’t own more money than received or minted.
3. Transaction non-malleability
   Cannot manipulate transactions en route to ledger.
   (Requires further changes to the construction.)

Implementation



Network simulation

third-scale Bitcoin network on EC2

Bitcoind + Zerocash hybrid currency

`libzerocash`

provides DAP interface

Statement for zkSNARK

Hand-optimized

`libsnark`

zkSNARK

SCIPR LAB

Instantiate Zerocash primitives and parameters

`bitcoind.`

Performance (quadcore desktop)

`Setup`	<2 min, 896MB params
`Mint`	23 μs 72B transaction
`Pour`	46 s, 1KB transaction
`Verify Transaction`	<9 ms/transaction
`Receive`	<2 ms/transaction

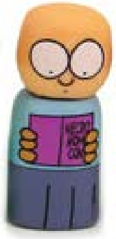
Trusted setup

- `Setup` generate fixed keys used by all provers and verifiers.
- If `Setup` is compromised at the dawn of the currency, attacker could later forge coins.
- Ran once. Once done and intermediate results erased, no further trust (beyond underlying cryptographic assumptions)
- Anonymity is unaffected by corrupted setup.
- Practical trustworthy protocol for running `Setup`?

Open research problems

- zkSNARKs can enforce policies and regulation in a privacy-preserving, corruption-proof way.
  - What policies are desireable and feasible?

I'm using up a coin with value v (unique) sn, and I know $r', r'', \rho, a_{\text{pk}}$ that are consistent with cm and I paid 10% tax and put my name in escrow with an authorized notary.



- Other Bitcoin applications
  - Blockchain compression
  - Turing-complete scripts/contracts
  - Proof of reserve

- Eliminating trusted setup.