Decentralized Anonymous Credentials and Electronic Payments from Bitcoin

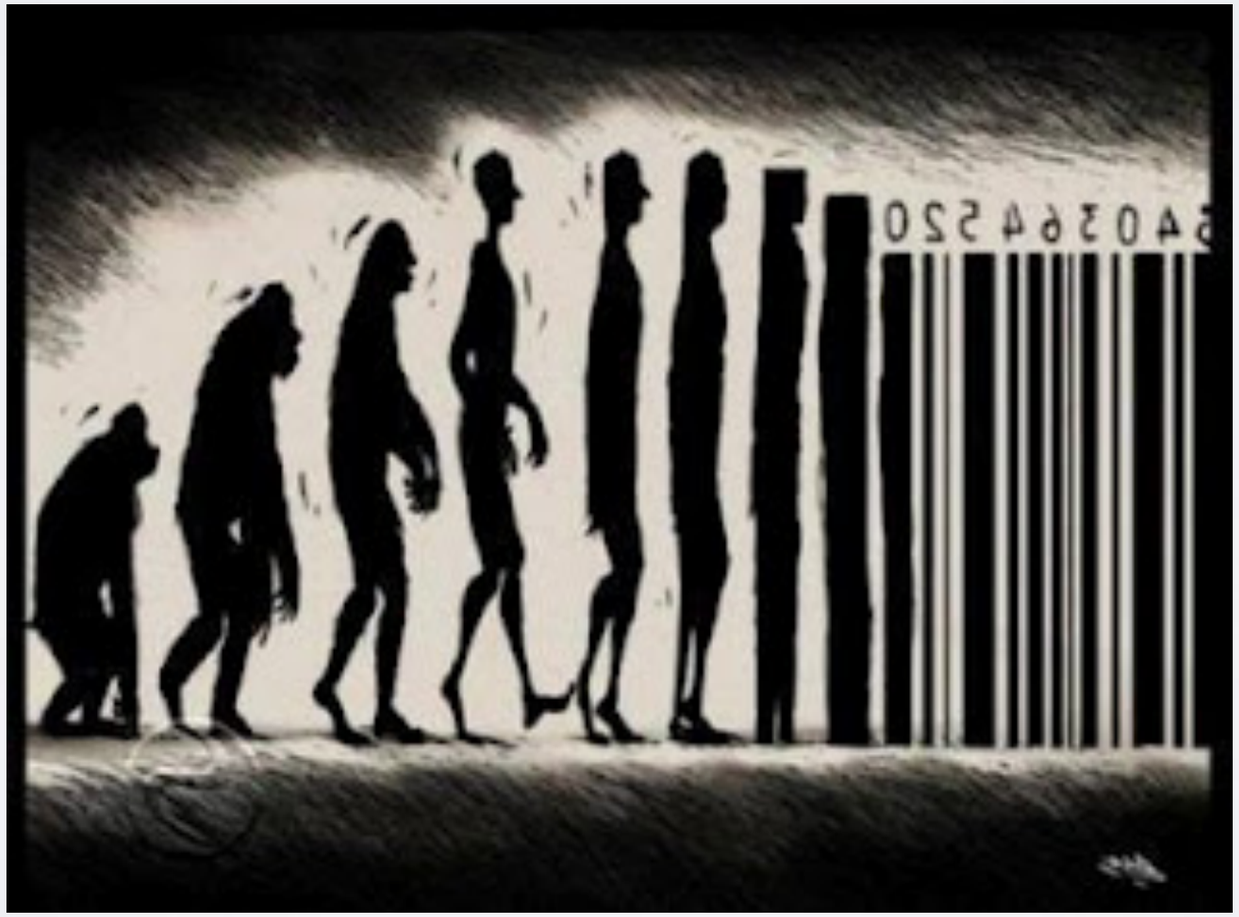
Matthew Green
Johns Hopkins University

Background

- **A bit about myself**
  - Researcher at Johns Hopkins University, focus on:
    - privacy (obliviosu transfer)
    - and apply
    - They can create currency steal

Problem: electronic cash

1) Very simple
2) Very difficult




Harrah's Council Bluffs
1 Harrah's Blvd. Council Bluffs, IA 51501
CASHOUT TICKET
VALIDATION 51-0601-7245-2982-3145
07/31/2010 18:13:16 TICKET # 7680
ZERO DOLLARS AND SEVENTEEN CENTS
$0.17
Ticket Void after 90 days ASSET# 1724

Problems

- **Centralization & Trust**
  - You traditionally need a trusted party to operate the bank
  - They can create currency, steal or simply fail

Problem: Privacy

- **Centralization & Trust**
  - You need a trusted party to operate the bank
  - They can create currency, steal or simply fail
- **Privacy**
  - The bank sees every transaction you make!

Problems

- **Centralization & Trust**
  - You need a trusted party to operat
  - They can create currency, steal or si
- **Privacy**
  - The bank sees every transaction you make!

Most academic literature focuses on this problem! Chaum82….

Problems

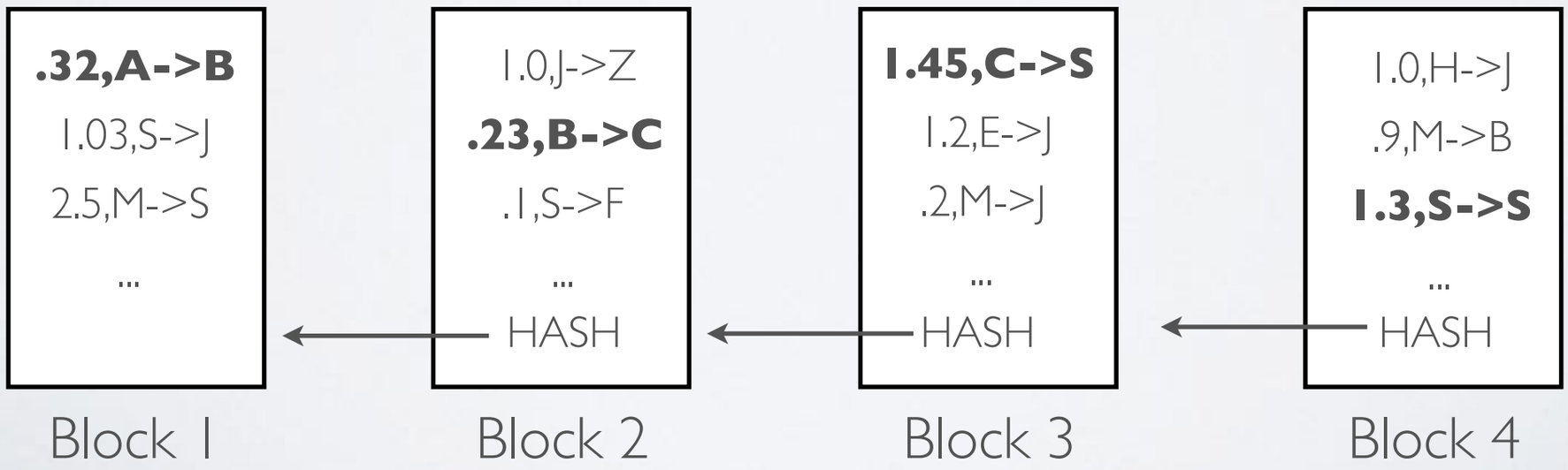
- **Centralization & Trust**
  - You need a trusted party to operate the bank
  - They can create currency, steal or simply fail
- **Privacy**
  - The bank sees every transaction you make!

In fact *this* appears to have been the more interesting problem

Bitcoin

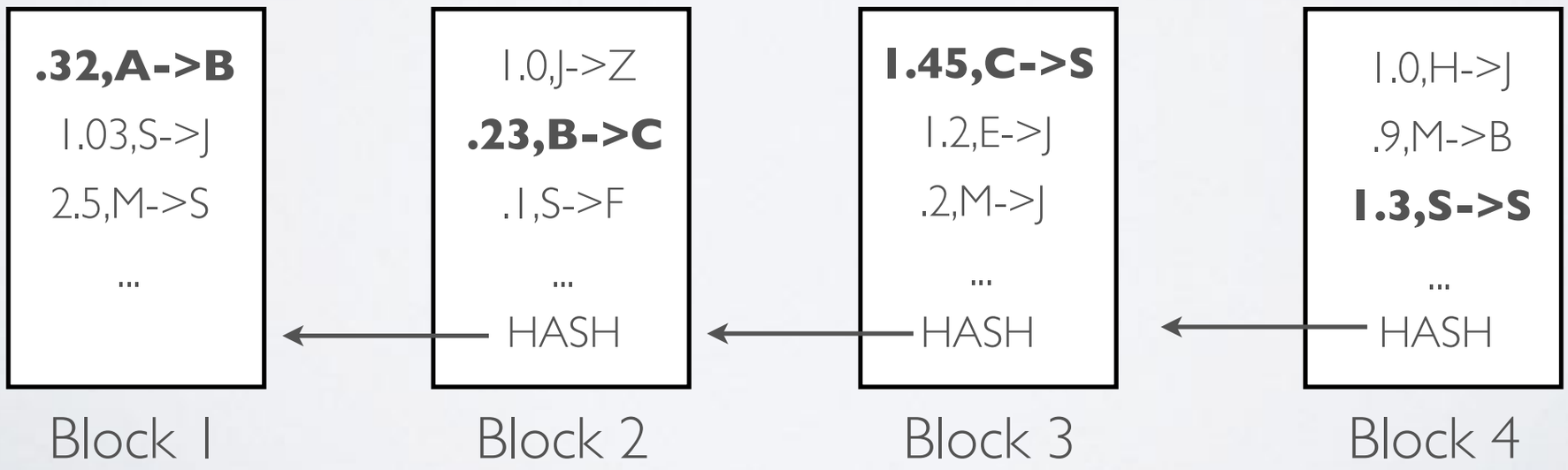
Bitcoin (Nakamoto '09)

- Bitcoin's core innovation is the block chain
  - A **decentralized append-only ledger** (divided into 'blocks' of many transactions)
- Massively replicated
- Everyone can download and see transactions



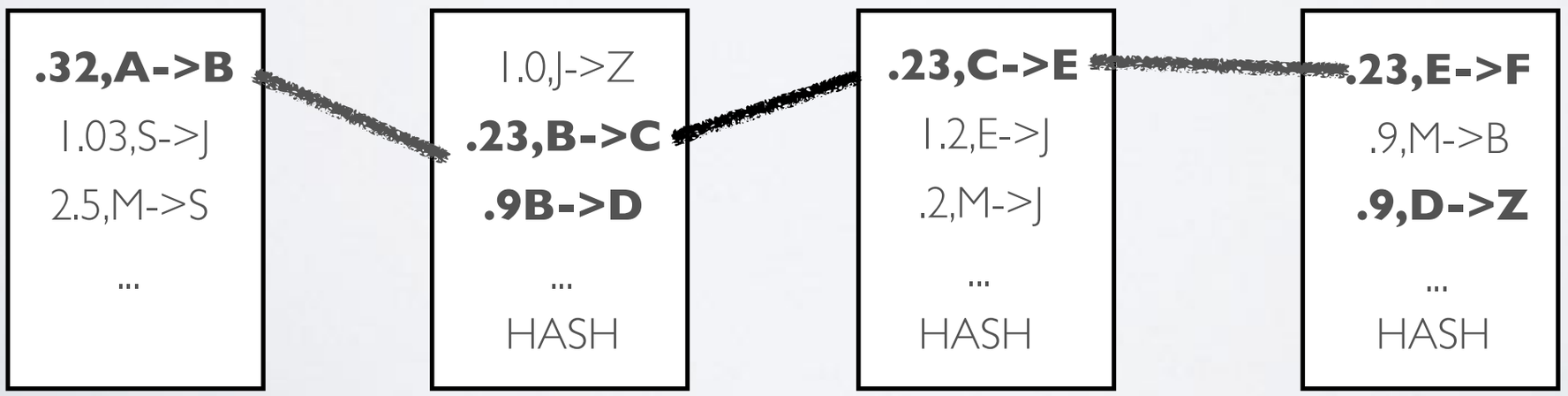
Bitcoin (Nakamoto '09)

- Bitcoin's core innovation is the block chain
  - Transactions are distributed via a P2P network
  - Miners select transactions, compete to solve PoWs; winner adds next block to a hash chain (tree)
- Solving a block 'creates' currency / transaction fees



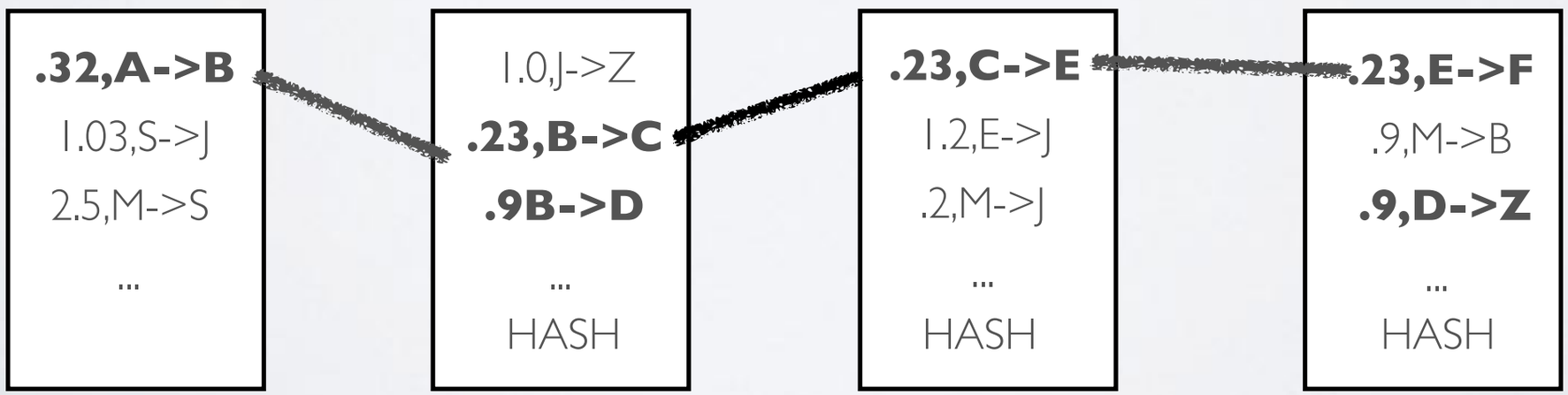
Bitcoin (Nakamoto '09)

- Using the ledger parties "write checks" to one another
  - User addresses are **public keys**
  - Standard transactions consist of:
    - A list of 'input transaction' IDs
    - A list of 'output addresses' and values
    - Signature(s)

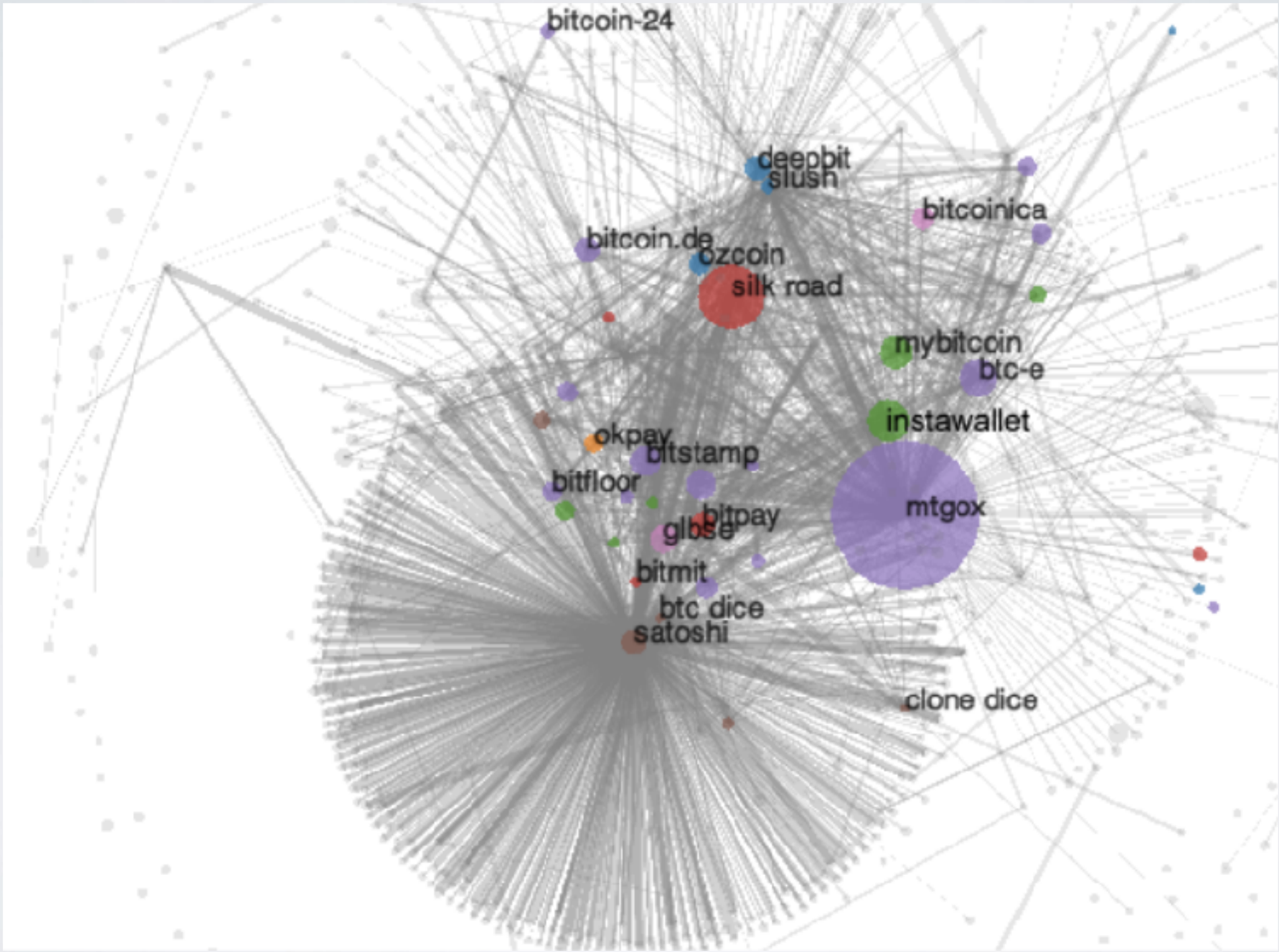


Bitcoin & Privacy

- **TL;DR: Bitcoin is not very anonymous**
  - Bitcoin transactions are recorded in a *public* ledger
  - Parties 'write checks' using pseudonyms (addresses)
  - If people can link you to your address, you're hosed
  - You're probably hosed (MPJLMVS13, RS14)

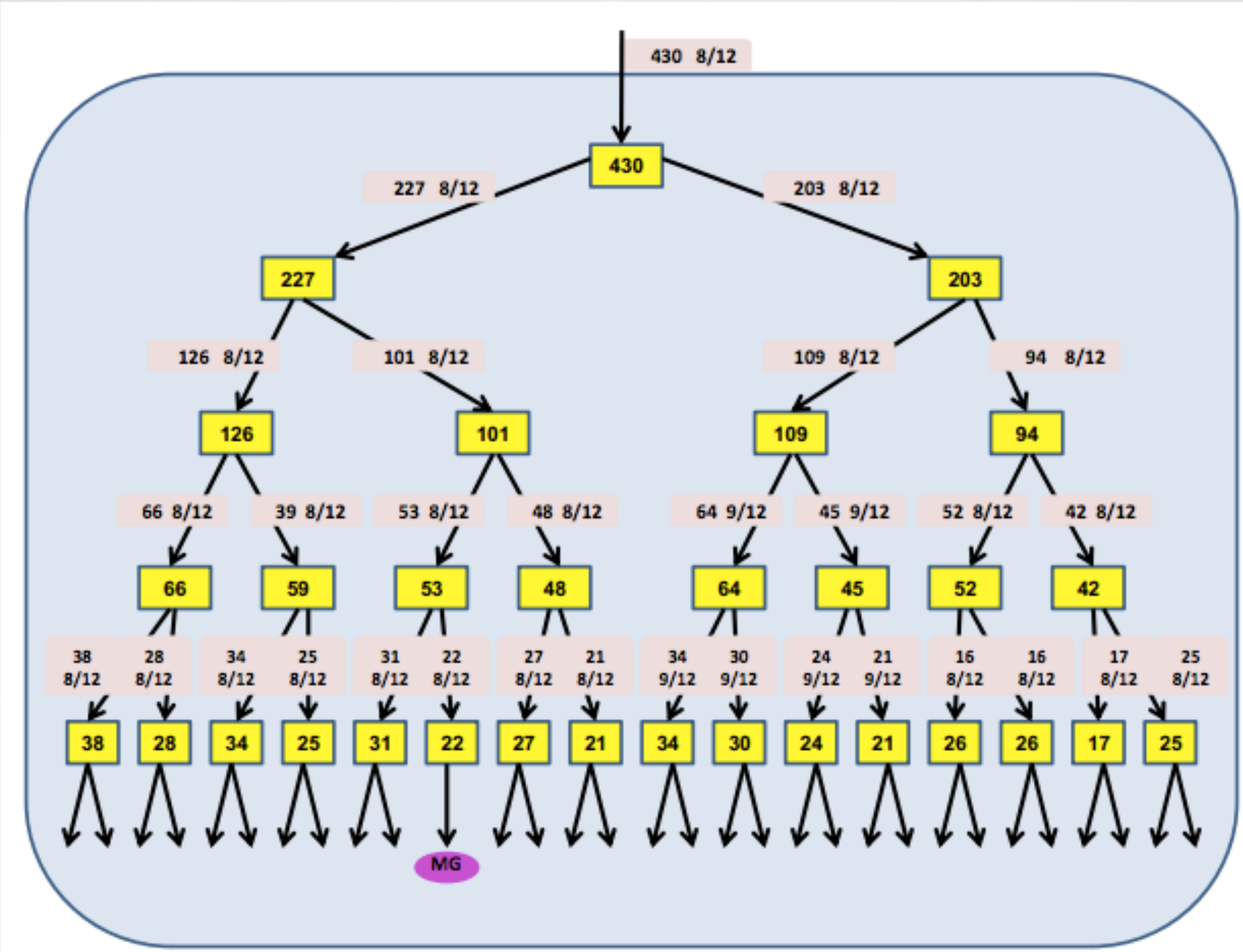


Bitcoin & Privacy



Source: MPJLMVS13

Bitcoin & Privacy



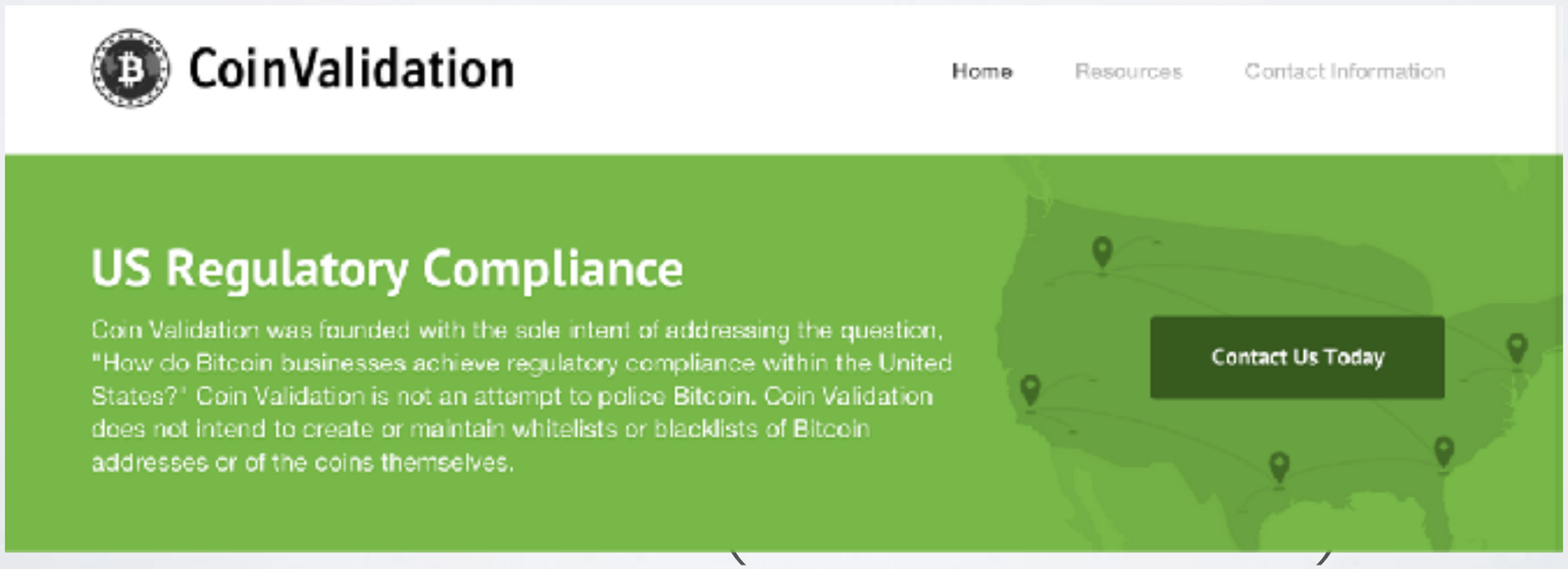
Source: RS14

Today's privacy solutions

- "Be careful"
- CoinJoin (mix with friends)
- Use 'laundry' services
  - Mix many users' coins together
- You must really trust the laundry

This matters!

- Solving the privacy problem is crucial to Bitcoin's long-term success
- Existing countermeasures don't address the problem, and probably never will
- A real solution may yield useful new techniques

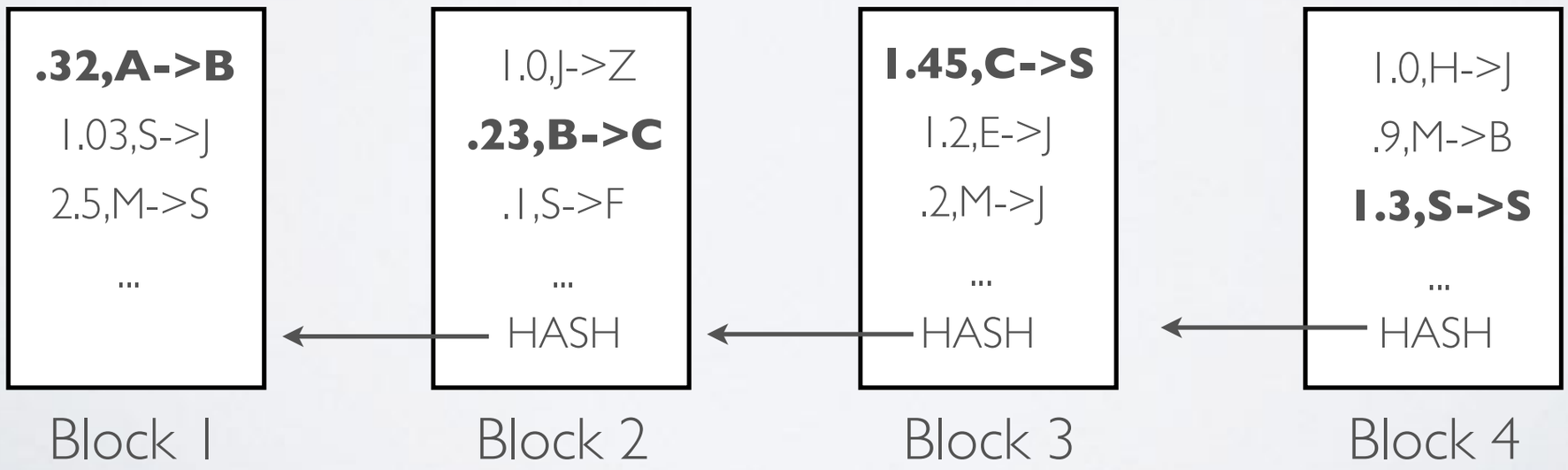


Outline of this talk

- Today I'm going to talk about two "fixes" for this problem & a neat side application:
  - Zerocoin - privacy for Bitcoin
  - Zerocash - *decentralized anonymous payments* for Bitcoin
  - Decentralized Anonymous Credentials

Bitcoin in 2 slides (1/2)

- Bitcoin's core innovation is the block chain
  - A **decentralized append-only ledger** (divided into 'blocks' of many transactions)
  - Massively replicated
  - The blocks are connected through hash chaining



Zerocoin

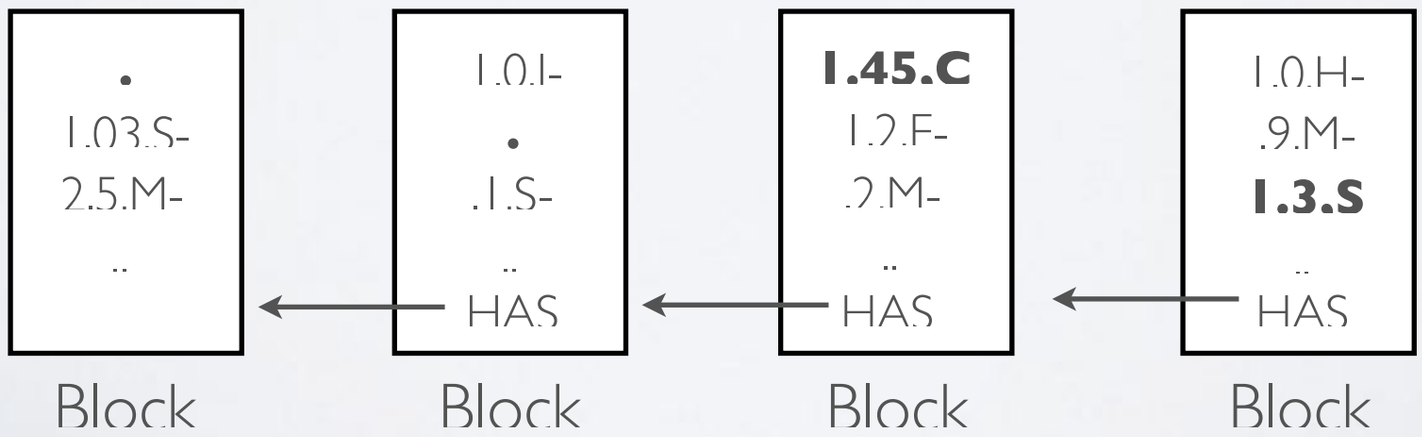
Joint work with
Ian Miers, Christina Garman, Avi Rubin (*Oakland '13*)

Let's use e-Cash for Bitcoin!

- e-Cash due to Chaum [82] (many subsequent works)
  - Untraceable electronic cash
  - Traditional schemes withdraw 'coins' from a **central bank** (using **blind signatures**)
- **Not compatible with the Bitcoin security model**

Zerocoin

- New approach to creating electronic coins
  - Based on a technique due to Sander and Ta-shma
  - Extends Bitcoin by adding a ‘decentralized laundry’
- **No bank:** Requires only a trusted bulletin board
- **Bitcoin block chain gives us this ‘for free’!**



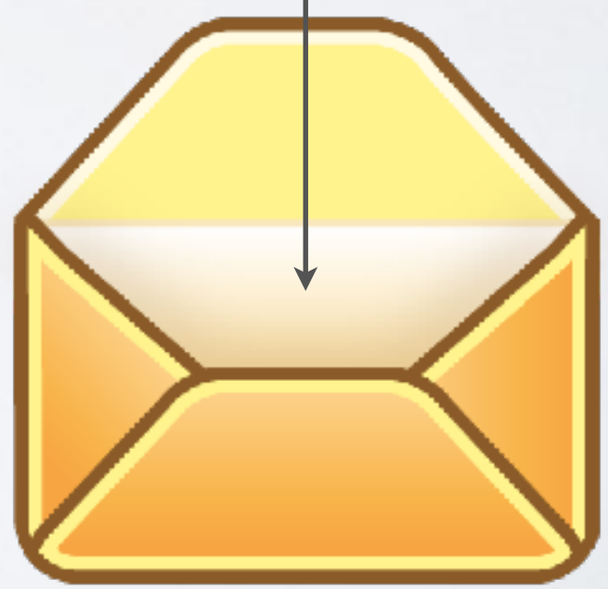
The high level idea

- I can take Bitcoin from my wallet
  - Turn them into ‘Zerocoins’
  - Where they get ‘mixed up’ with many other users’ coins
- I can redeem them to a new fresh Wallet

Minting Zerocoin

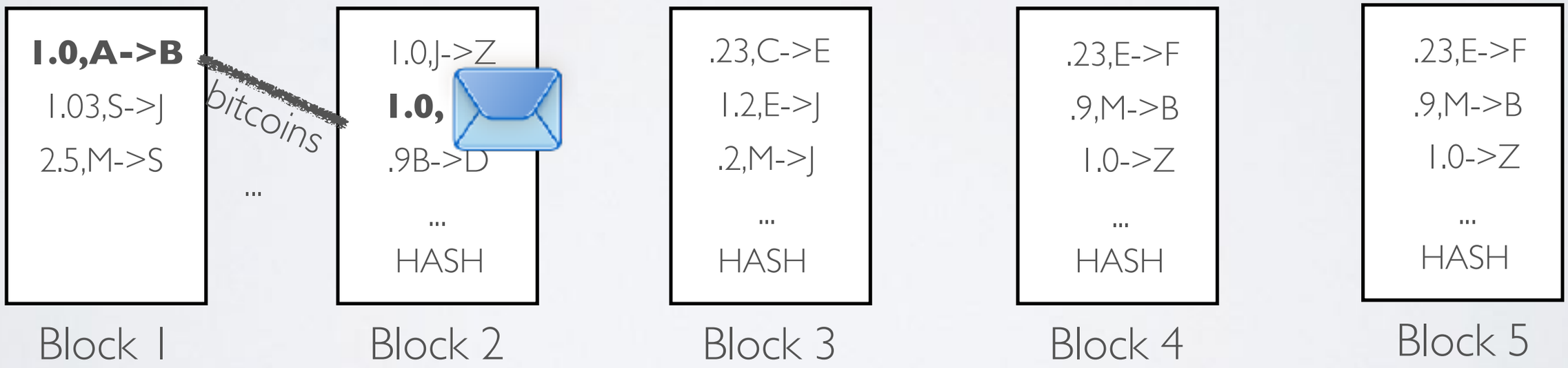
- Zerocoins are just numbers
  - Each is a digital commitment to a random serial number
  - Anyone can make one!

8238482734710I2983



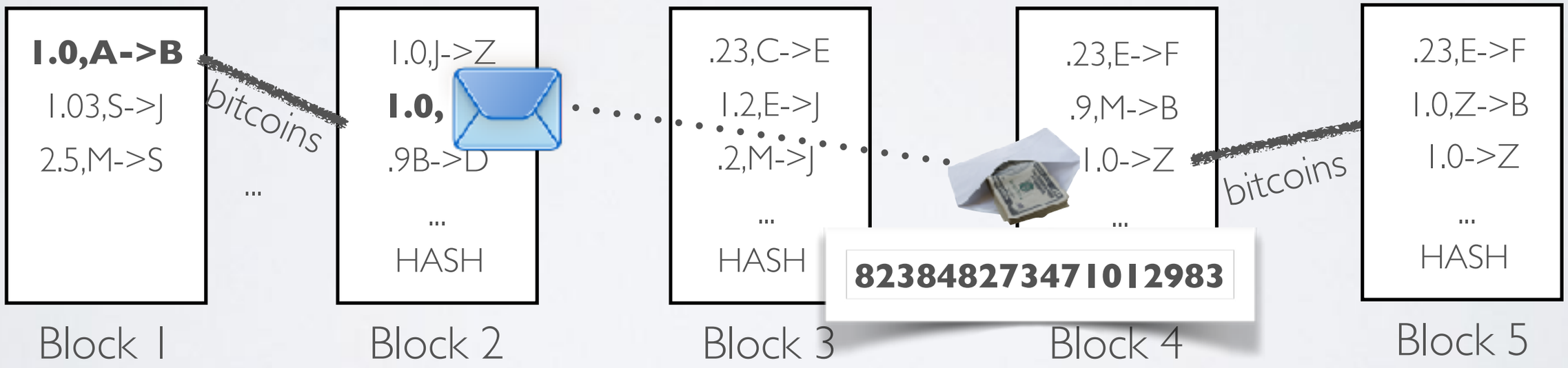
Minting Zerocoin

- Zerocoins are just numbers
  - They have value once you put them on the block chain
  - This costs e.g., 1 bitcoin



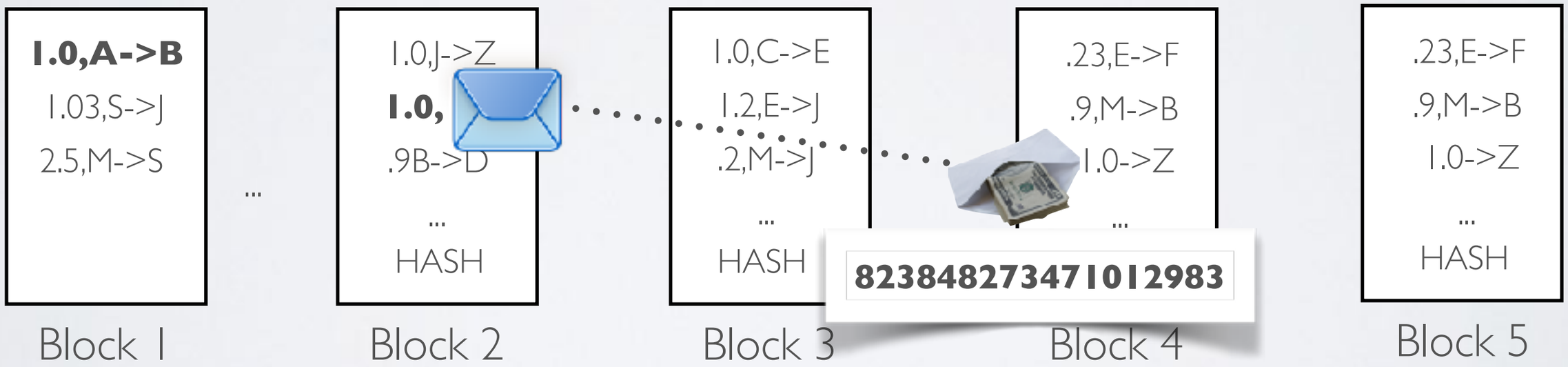
Redeeming Zerocoin

- You can redeem zerocoins back into bitcoins
  - Reveal the serial number &
    Prove that it corresponds to some Zerocoin on the chain
  - In exchange you get one bitcoin



Spending Zerocoin

- Why is spending anonymous?
  - It's all in the way we 'prove' we have a Zerocoin
  - This is done using a zero knowledge proof



Spending Zerocoin

- Zero knowledge [Goldwasser, Micali 1980s, and beyond]
  - Prove a statement without revealing any other information
  - Here we prove that:
    (a) there exists a Zerocoin in the block chain
    (b) we just revealed the actual serial number inside of it
- Revealing the serial number prevents double spending
- The trick is doing this efficiently!

Spending Zerocoin

- Our approach
  - Use an efficient RSA one-way accumulator
  - Accumulate $C_1, C_2, \ldots, C_N$ to produce accumulator A
  - Then prove knowledge of a witness s.t. $C \in inputs(A)$
  - And prove knowledge that C opens to the serial number

Requires a DDL proof (**~25kb**)
for each spend. In the block chain.

Spending Zerocoin

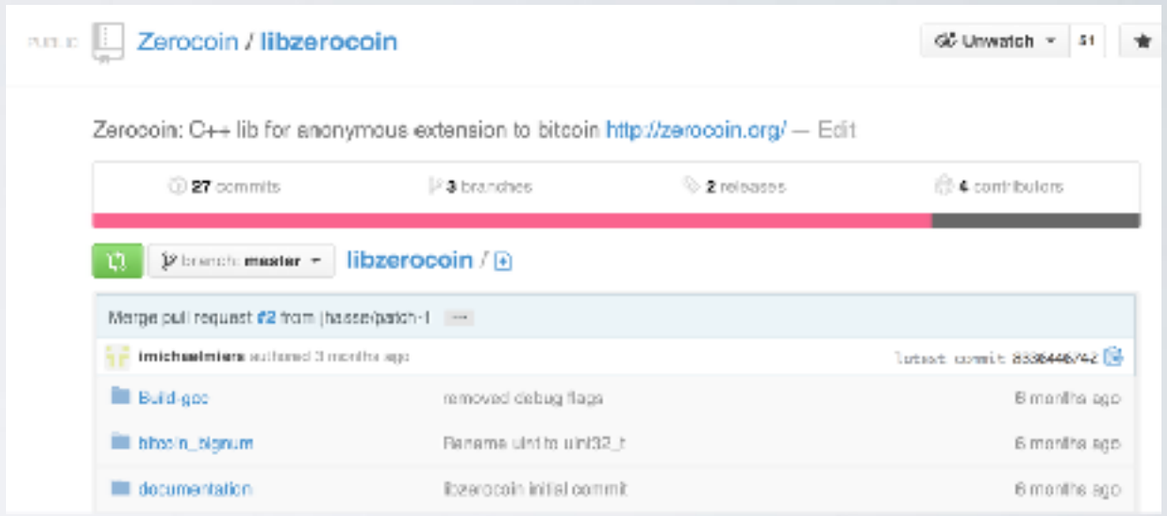
- Our approach
  - Use an efficient one-way accumulator
  - Accumulate $C_1, C_2, \ldots, C_N$ to produce accumulator A
  - T
  - And prove knowledge that opens to the serial number

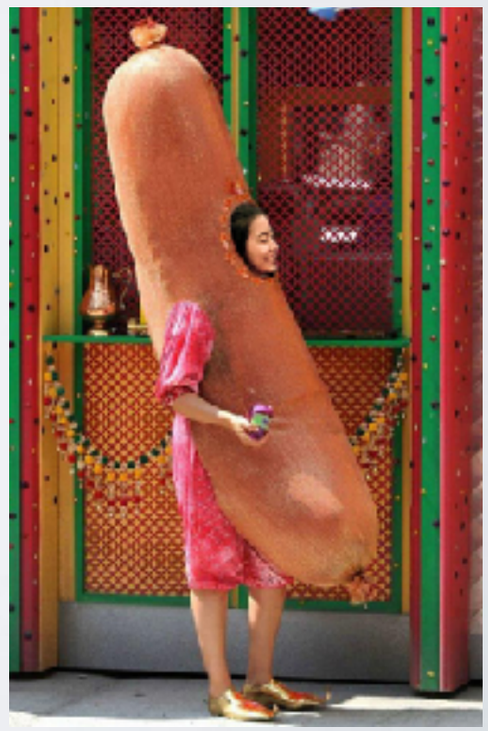
This is a problem for the Bitcoin community! *****

Requires a DDL proof (**~25kb**) for each spend. In the block chain.

Summary of Zerocoin

- Good first approach:
  - Implemented!
  - Proofs are (too?) big
- Coins all have the same value
- Must convert ‘zerocoins’ to ‘bitcoins’ in order to spend them





Summary of Zerocoin

- (

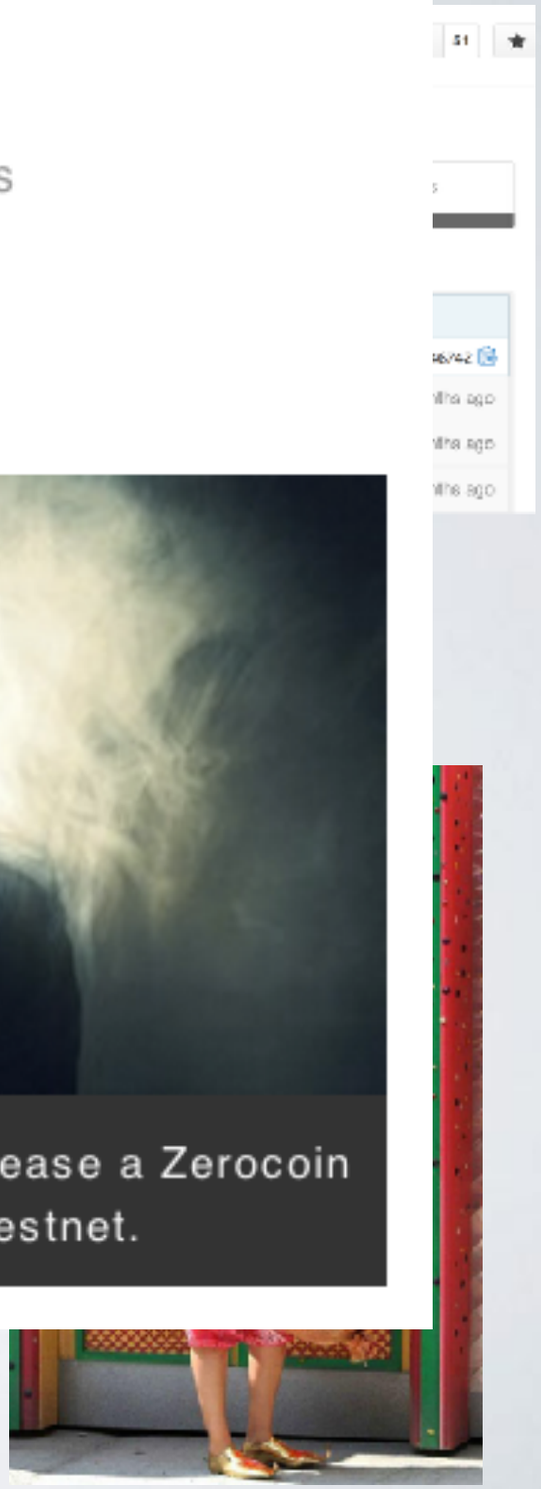
Anoncoin to Release Zerocoin Implementation on Testnet This Week

Kyle Torpey 07/04/2014 Altcoins, Bitcoin, News 2 Comments

Posted 1 day ago

Many people in the Bitcoin community have been fascinated by the idea of a relatively new project called Zerocoin. Although the Bitcoin core developers don't seem like they're going to implement this feature for improve anonymity on the Bitcoin blockchain, that won't stop other altcoins from doing just that. The original team behind the

Anoncoin is almost ready to release a Zerocoin implementation on a testnet.



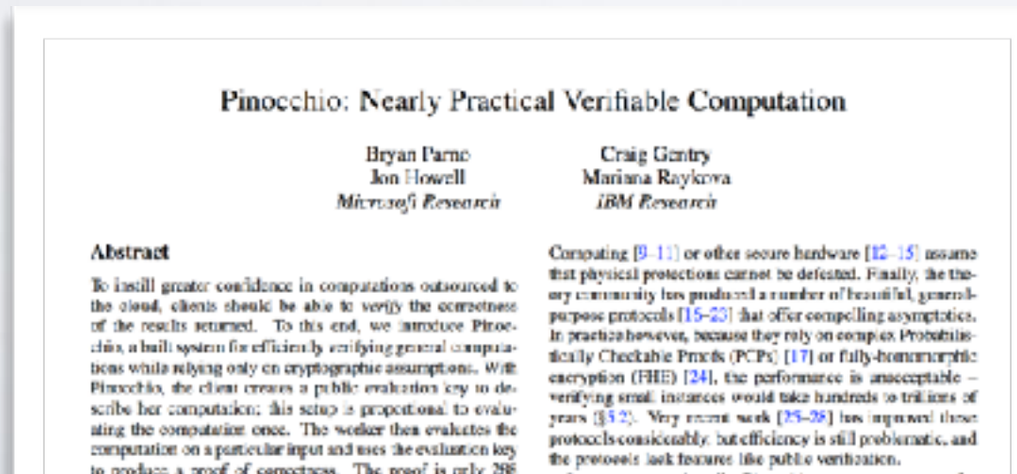
Zerocash

Joint work with

- Alessandro Chiesa, Madars Virza, Ian Miers, Christina Garman, Eran Tromer, Eli Ben-Sasson (*Oakland '14)*

A better tool

- Better, smaller 'proofs' of knowledge:
  - **S**uccinct **N**on-Interactive **AR**guments of **K**nowledge (zkSNARKs) (Parno *et al.*, Ben-Sasson *et al.)*
- 288 byte proof for arbitrary-sized arithmetic circuits
- And there are C compilers!



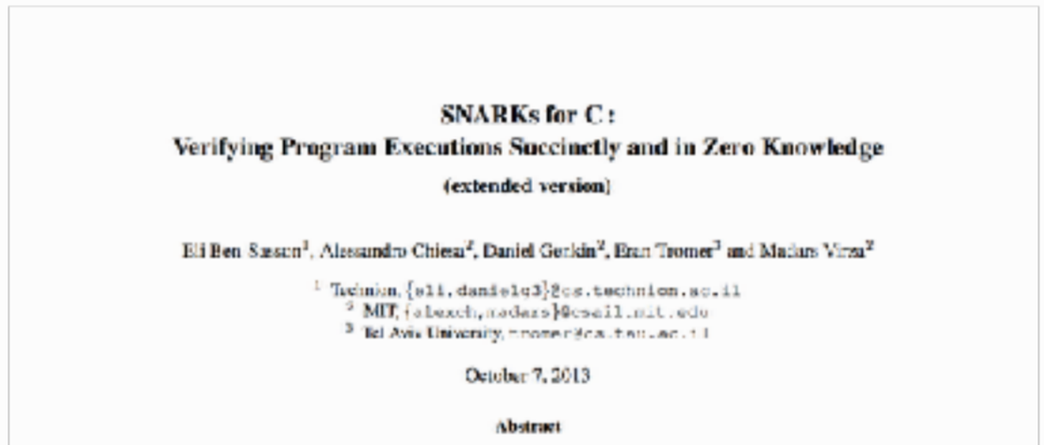
Pinocchio: Nearly Practical Verifiable Computation

Bryan Parno, Jon Howell — *Microsoft Research*
Craig Gentry, Mariana Raykova — *IBM Research*

Abstract

To instill greater confidence in computations outsourced to the cloud, clients should be able to *verify* the correctness of the results returned. To this end, we introduce Pinocchio, a built system for efficiently verifying general computations while relying only on cryptographic assumptions. With Pinocchio, the client creates a public evaluation key to describe her computation; this setup is proportional to evaluating the computation once. The worker then evaluates the computation on a particular input and uses the evaluation key to produce a proof of correctness. The proof is only 288

Computing [9–11] or other secure hardware [12–15] assume that physical protections cannot be defeated. Finally, the theory community has produced a number of beautiful, general-purpose protocols [15–23] that offer compelling asymptotics. In practice however, because they rely on complex Probabilistically Checkable Proofs (PCPs) [17] or fully-homomorphic encryption (FHE) [24], the performance is unacceptable – verifying small instances would take hundreds to trillions of years (§ 2). Very recent work [25–28] has improved these protocols considerably, but efficiency is still problematic, and the protocols lack features like public verification.



SNARKs for C:
Verifying Program Executions Succinctly and in Zero Knowledge
(extended version)

Eli Ben Sasson[1], Alessandro Chiesa[2], Daniel Genkin[2], Eran Tromer[3] and Madars Virza[2]

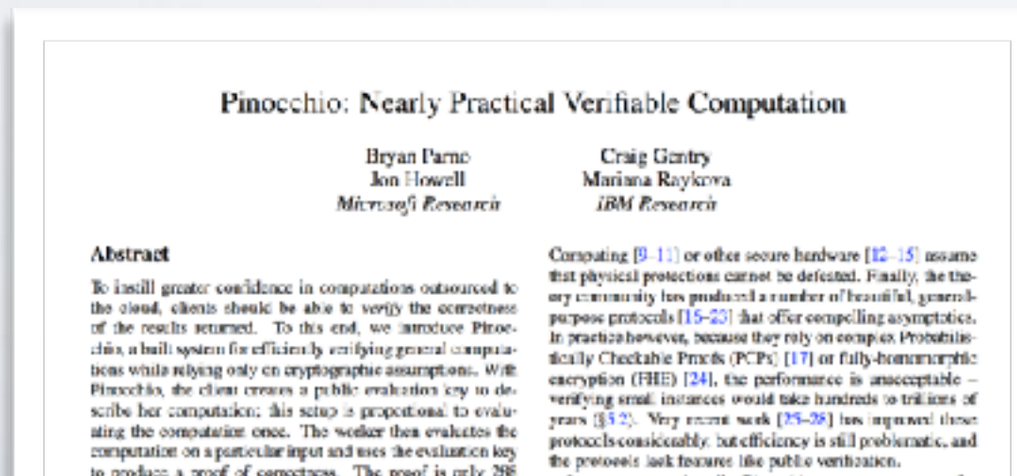
[1] Technion, {eli, danielg3}@cs.technion.ac.il
[2] MIT, {alexch, nadias}@csail.mit.edu
[3] Tel Aviv University, tromer@cs.tau.ac.il

October 7, 2013

Abstract

How <u>not</u> to use SNARKs

- In theory this should be simple:
  - We've already coded up Zerocoin in C++
  - Let's run our existing software through the zkSNARK compilers to get small proofs
- <u>Surprise</u>: This gives *large*, *impractical* circuits (proving takes a long time)



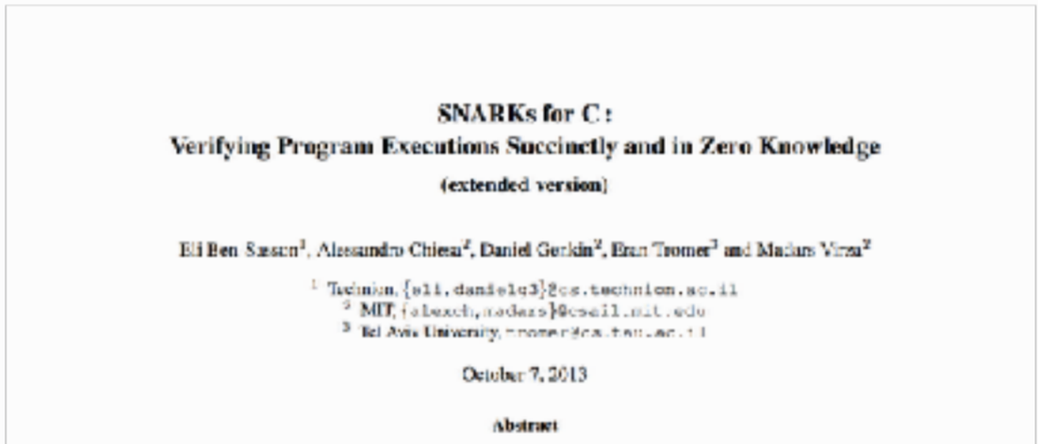
Pinocchio: Nearly Practical Verifiable Computation

Bryan Parno, Jon Howell — *Microsoft Research*
Craig Gentry, Mariana Raykova — *IBM Research*

Abstract

To instill greater confidence in computations outsourced to the cloud, clients should be able to verify the correctness of the results returned. To this end, we introduce Pinocchio, a built system for efficiently verifying general computations while relying only on cryptographic assumptions. With Pinocchio, the client creates a public evaluation key to describe her computation; this setup is proportional to evaluating the computation once. The worker then evaluates the computation on a particular input and uses the evaluation key to produce a proof of correctness. The proof is only 288

Computing [9–11] or other secure hardware [12–15] assume that physical protections cannot be defeated. Finally, the theory community has produced a number of beautiful, general-purpose protocols [15–23] that offer compelling asymptotics. In practice however, because they rely on complex Probabilistically Checkable Proofs (PCPs) [17] or fully-homomorphic encryption (FHE) [24], the performance is unacceptable – verifying small instances would take hundreds to trillions of years (§ 2). Very recent work [25–28] has improved these protocols considerably, but efficiency is still problematic, and the protocols lack features like public verification.



SNARKs for C:
Verifying Program Executions Succinctly and in Zero Knowledge
(extended version)

Eli Ben Sasson[1], Alessandro Chiesa[2], Daniel Genkin[2], Eran Tromer[3] and Madars Virza[2]

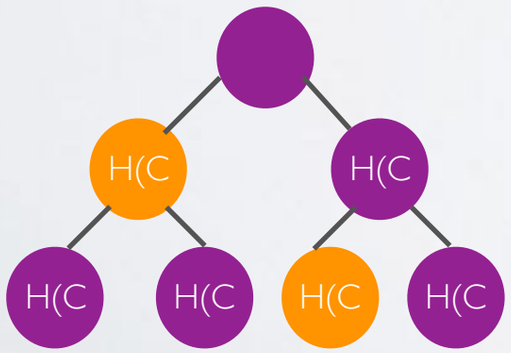
[1] Technion, {eli,danielg}@cs.technion.ac.il
[2] MIT, {alexch,madars}@csail.mit.edu
[3] Tel Aviv University, tromer@cs.tau.ac.il

October 7, 2013

Abstract

How to use SNARKs

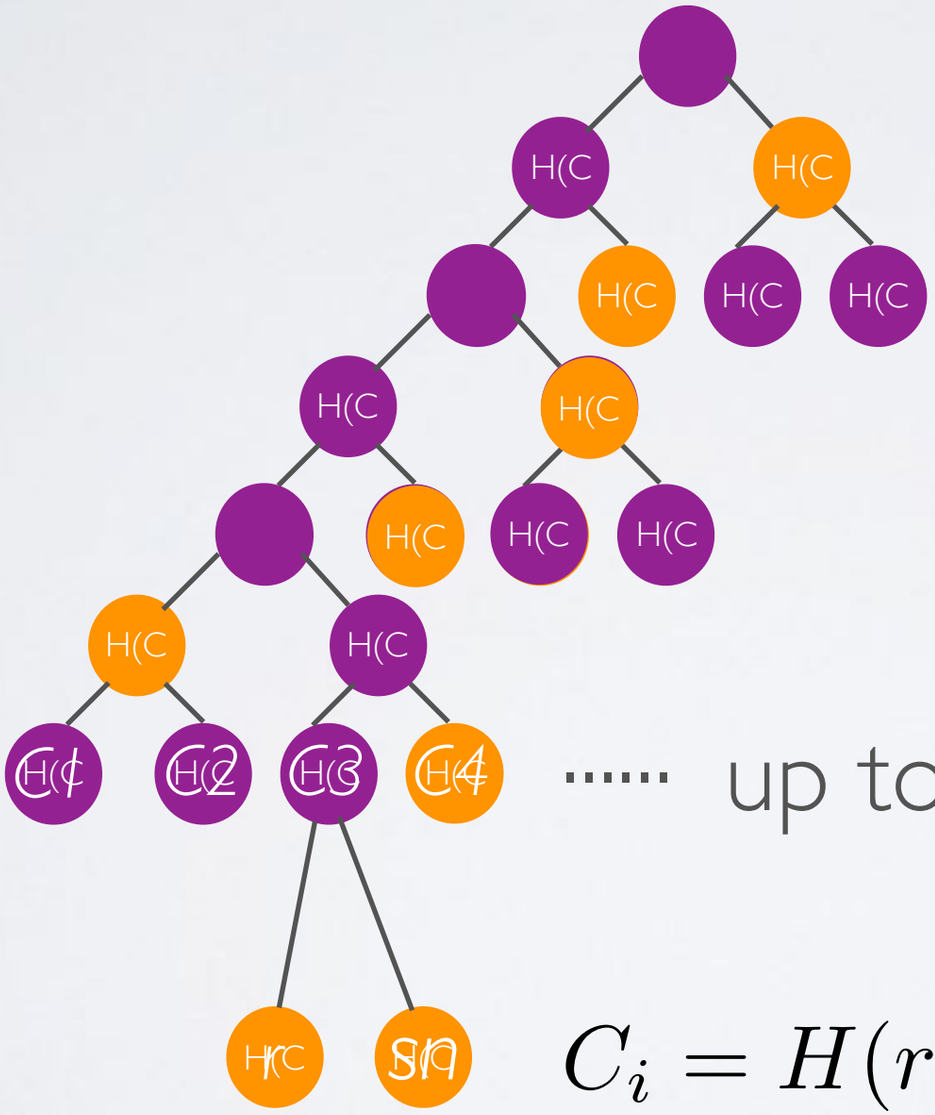
- Start from scratch:
  - Develop an entirely new construction with small circuits
  - Modify Zerocoin to use hash functions for commitments, hash trees for an accumulator (SHA256 for all hashes)
- Hand-optimize everything



C_{SHA256} (circuit for SHA256)	Gate count
Message schedule	8 032
All rounds	21 632
1 round (of 64)	338
Finalize	288
Total	29 952

Figure 2: Size of circuit C_{SHA256} for SHA256.

Proposed Zerocash tree



..... up to 2^{64} coins

$$C_i = H(r_i||sn)$$

But wait a second...

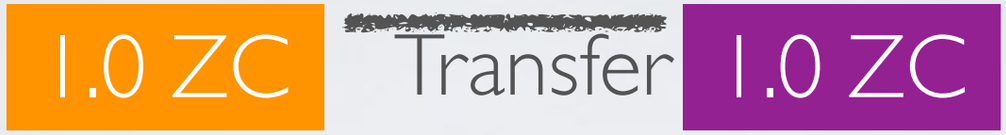
- If the proofs are powerful & efficient, why do we need Bitcoin anymore?
  - Let's add hidden values to the coin: $C_i = H(r_i||v||sn)$
  - Create transactions to split/merge coins
- Allow payments (from Alice to Bob) that don't reveal value
  - Pay to individuals, pay to address

To split a coin:

1. "Spend" the input coin
(by revealing its serial number)
2. "Mint" two new coins
3. Prove that the new coins total to
the value of the first coin

To merge two coins:

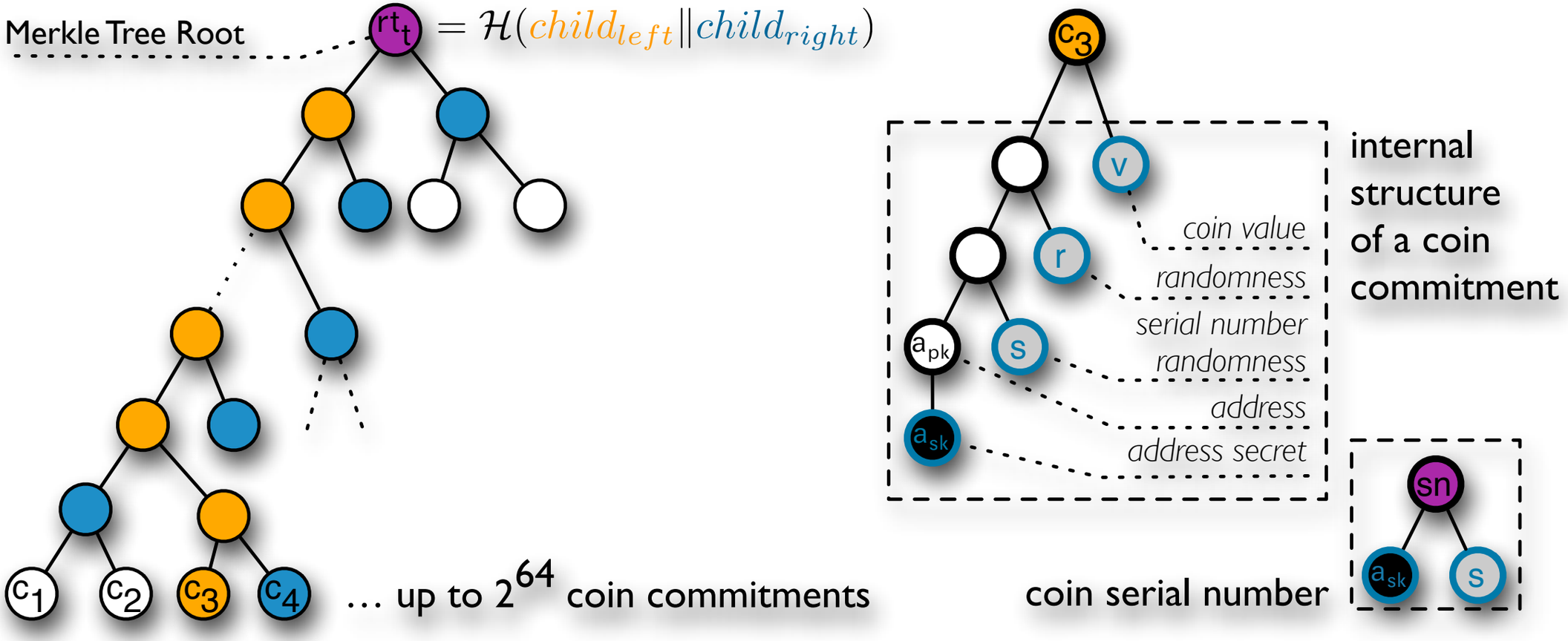
1. "Spend" the input coins
(by revealing their serial numbers)
2. "Mint" a new coin
3. Prove that the old coins total to
the value of the new coin



To pay a coin:

1. Transfer the coin secrets to the target user
2. Embed the recipient's 'address' A = H(x)
3. User must prove knowledge of x to redeem

Result: Zerocash



Properties of zkSNARKs

- Turns out that zkSNARKs may (or may not) be non-malleable
- Our assumptions assume only soundness & ZK
- In practice, we use MACs + OTS to 'build' non-malleability into our transaction format
- Todo: simulation soundness

Result: Zerocash

- An fully untraceable, divisible electronic cash system
  - Coins are anonymous starting from Coinbase transaction
  - Coins can be split/joined ('poured'), paid and revealed
  - The only place where coin values need be public is when we offer transaction fees

Performance

	Proving time	**Proof Size**	**Verif. time**
Split	87 sec	288 bytes	8.6 ms
Merge	178 sec	288 bytes	8.6 ms

128-bit security level, single core i7 @ 2.7 GHz

So what's the catch?

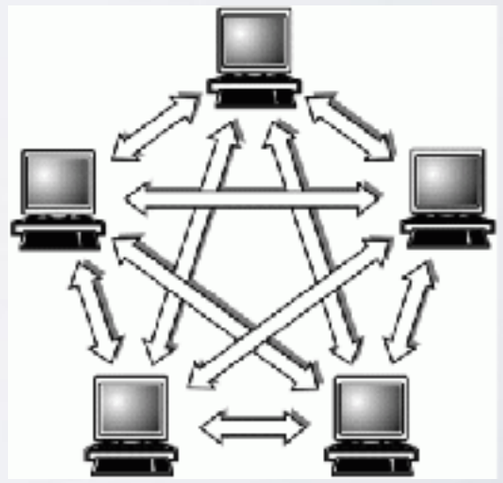
- The public parameters are quite large
  - **About 1.2 GB**
  - In context, that's about 7% the size of the blockchain
- They must be generated by a trusted party
  - A party who knows a trapdoor can forge proofs
  - But cannot de-anonymize transactions

The summary

- We now have efficient and fully anonymous e-Cash
  - With practical proving times & storage costs
  - A modestly irritating set of public parameters
- And code, which we will be releasing this summer

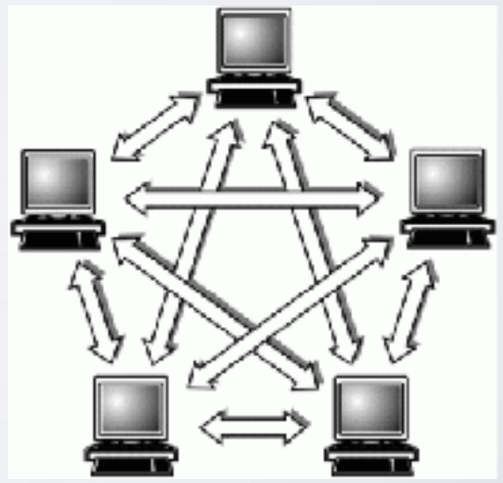
Anonymous Credentials

- Due to Chaum *et al.*
  - Allow us to prove statements about identity <u>without revealing it</u>
    - E.g., "I am an authorized user", "I am a subscriber"
    - Classic example: TPM anonymous attestation
- Usually requires a trusted anonymous credential issuer (e.g., TPM-DAA)



Anonymous Credentials!

- Observation: e-Cash is just a form of anonymous credential
  - New systems like Namecoin allow us to establish identities (with attributes, *e.g.*, time identity established)
  - By adding similar commitments to the identities/attributes we can prove statements about our identity
- No trusted credential issuer
- Can use this to implement decentralized anonymous reputation systems & 'subscription' services to manage resources in ad-hoc networks!



The future

- There's much more to talk about
  - Can something like this be deployed?
  - What are the ethics of doing it?
  - What's the future of Bitcoin as a technology? As a currency?
  - What about identity management?

The paper(s):

spar.isi.jhu.edu/~mgreen/ZerocoinOakland.pdf

https://eprint.iacr.org/2013/622.pdf

(Zerocash paper coming soon)

Code & project website:

zerocoin.org