Towards making ₿ anonymous

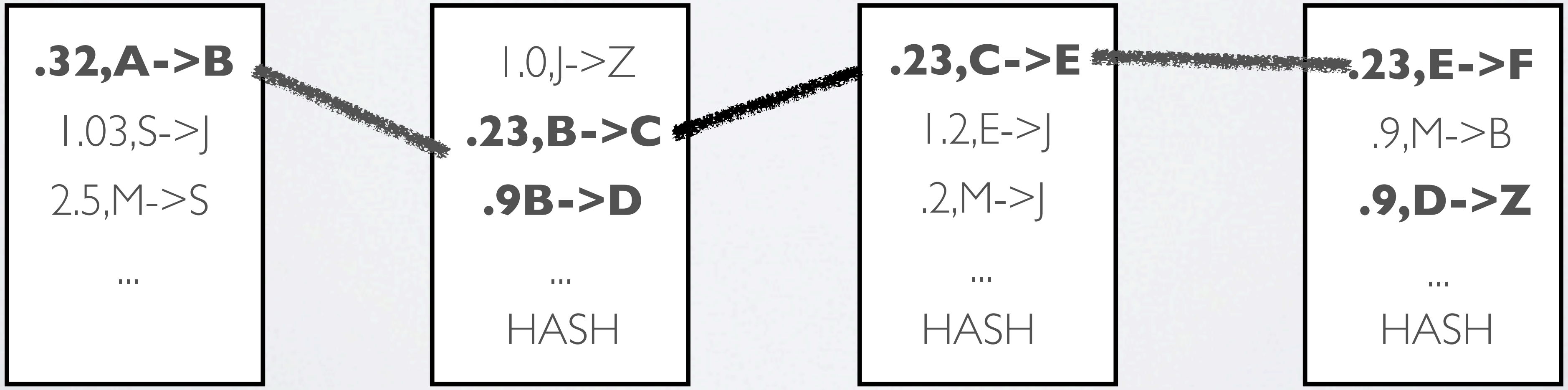
Matthew Green
Johns Hopkins University

Joint work with

- Ian Miers, Christina Garman, Avi Rubin (*Oakland '13*)
- Alessandro Chiesa, Madars Virza, Ian Miers, Christina Garman, Eran Tromer, Eli Ben-Sasson (*In submission)*

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 screwed
  - <u>You're probably screwed</u>



.32,A->B
1.03,S->J
2.5,M->S
...

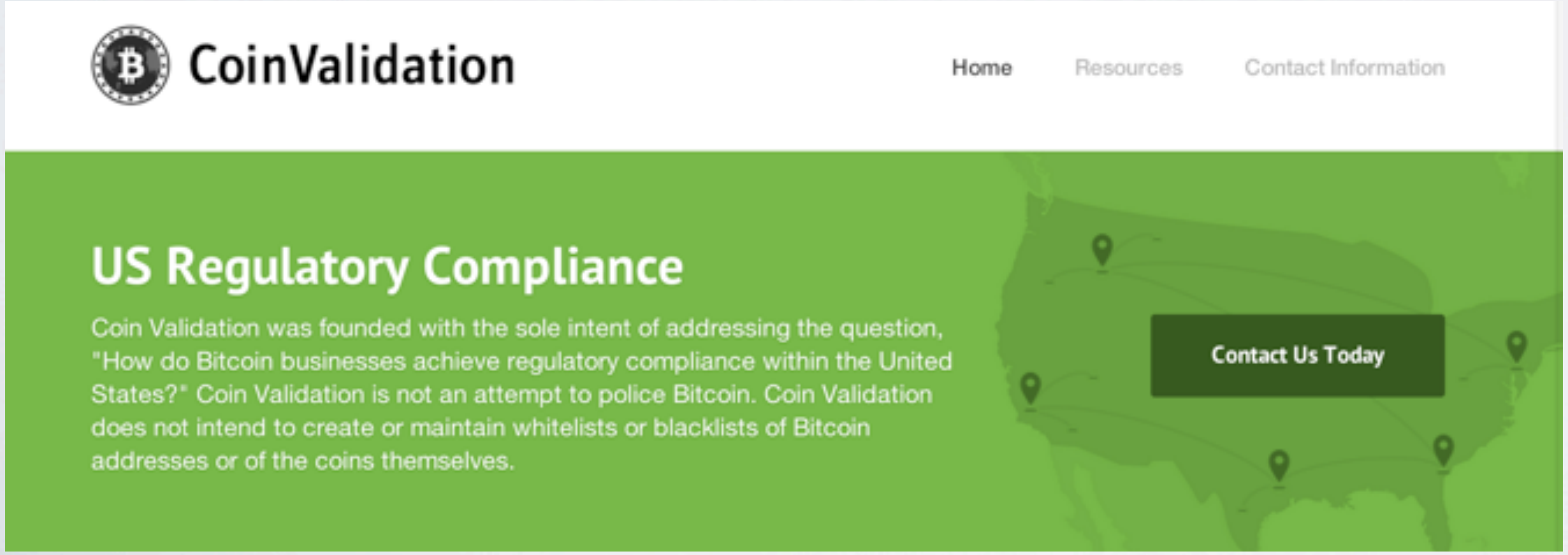
1.0,J->Z
.23,B->C
.9B->D
...
HASH

.23,C->E
1.2,E->J
.2,M->J
...
HASH

.23,E->F
.9,M->B
.9,D->Z
...
HASH

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



US Regulatory Compliance

Coin Validation was founded with the sole intent of addressing the question, "How do Bitcoin businesses achieve regulatory compliance within the United States?" Coin Validation is not an attempt to police Bitcoin. Coin Validation does not intend to create or maintain whitelists or blacklists of Bitcoin addresses or of the coins themselves.

Outline of this talk

- Today I’m going to talk about two “fixes” for this problem:
  - Zerocoin - privacy for Bitcoin
  - Zerocash - *efficient and deployable* privacy for Bitcoin

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**)

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**)

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

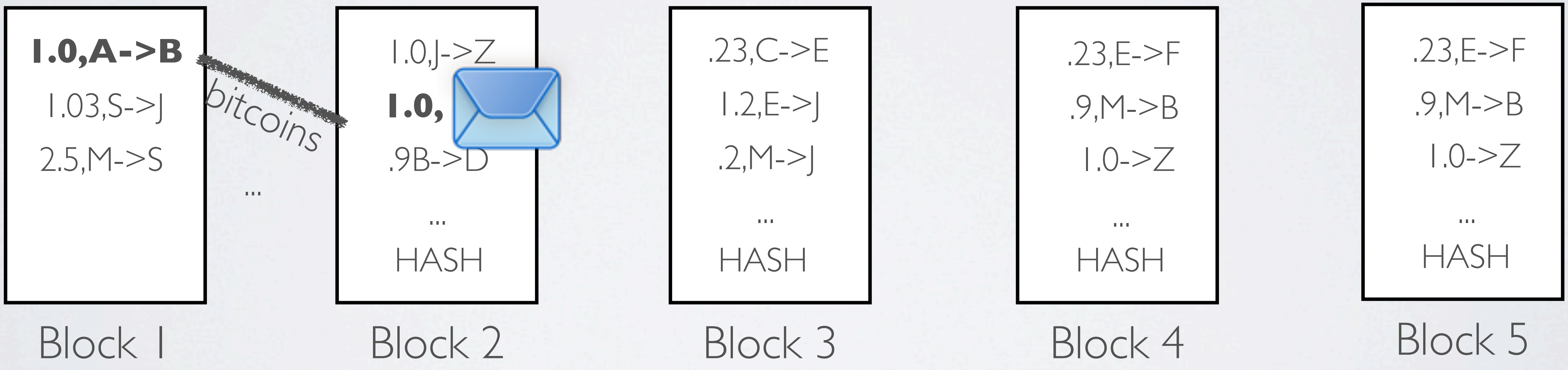
Minting Zerocoin

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

823848273471012983

Minting Zerocoin

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



1.0,A->B
1.03,S->J
2.5,M->S
Block 1

bitcoins

...

1.0,J->Z
1.0,
.9B->D
...
HASH
Block 2

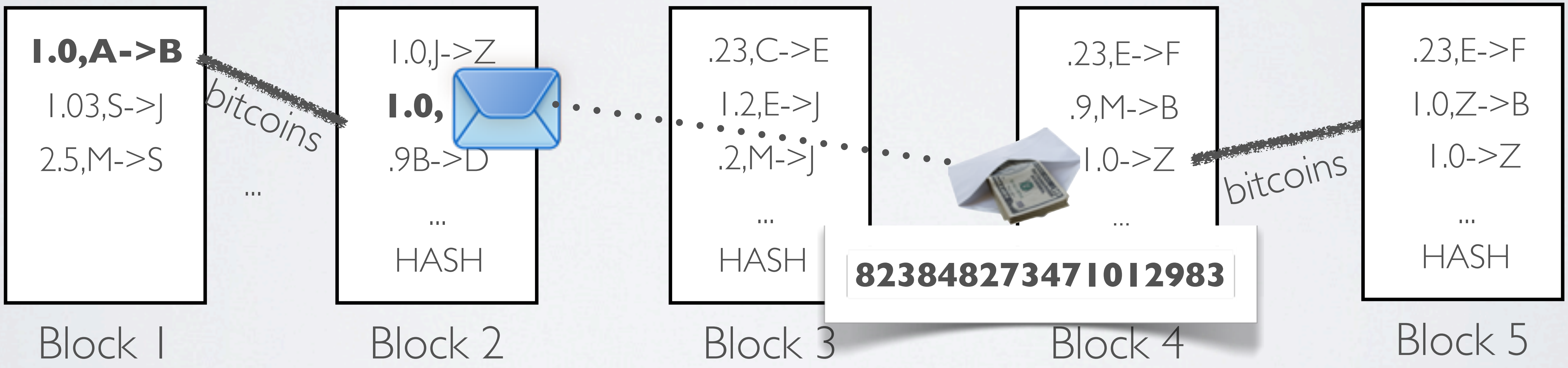
.23,C->E
1.2,E->J
.2,M->J
...
HASH
Block 3

.23,E->F
.9,M->B
1.0->Z
...
HASH
Block 4

.23,E->F
.9,M->B
1.0->Z
...
HASH
Block 5

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



1.0,A->B
1.03,S->J
2.5,M->S

Block 1

1.0,J->Z
1.0,
.9B->D
...
HASH

Block 2

.23,C->E
1.2,E->J
.2,M->J
...
HASH

Block 3

.23,E->F
.9,M->B
1.0->Z
...

Block 4

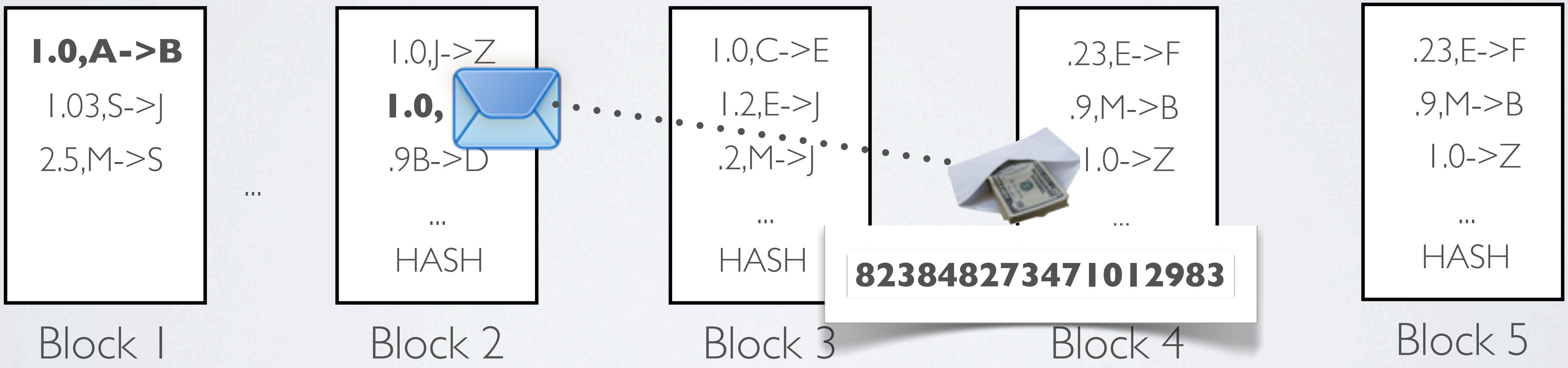
.23,E->F
1.0,Z->B
1.0->Z
...
HASH

Block 5

823848273471012983

Spending Zerocoin

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



1.0,A->B

1.03,S->J

2.5,M->S

Block 1

...

1.0,J->Z

1.0,

.9B->D

...

HASH

Block 2

1.0,C->E

1.2,E->J

.2,M->J

...

HASH

Block 3

.23,E->F

.9,M->B

1.0->Z

...

Block 4

823848273471012983

.23,E->F

.9,M->B

1.0->Z

...

HASH

Block 5

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.

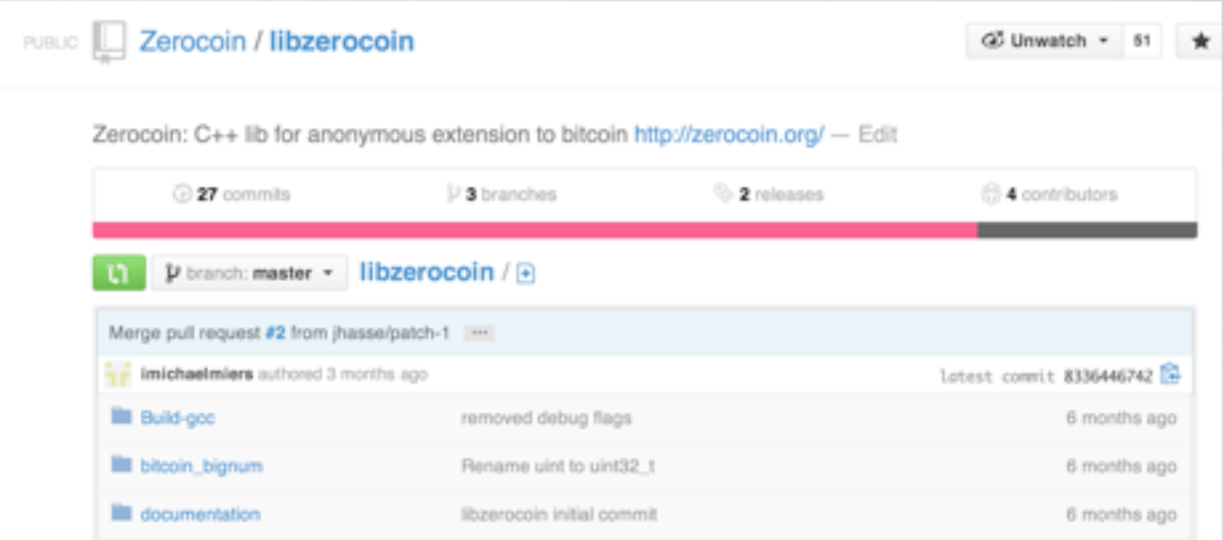
- Ou
  - U
  - A
  - T

Response from 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



Zerocoin / libzerocoin

Zerocoin: C++ lib for anonymous extension to bitcoin http://zerocoin.org/ — Edit

27 commits | 3 branches | 2 releases | 4 contributors

libzerocoin /

Merge pull request #2 from jhasse/patch-1

imichaelmiers authored 3 months ago — latest commit 8336446742

Build-gcc	removed debug flags	6 months ago
bitcoin_bignum	Rename uint to uint32_t	6 months ago
documentation	libzerocoin initial commit	6 months ago

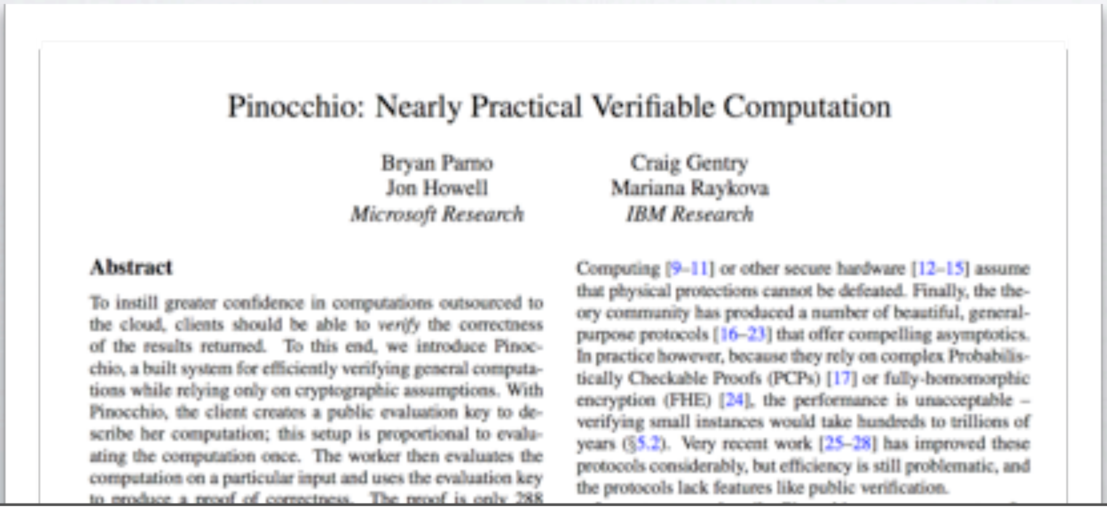
Zerocash

Joint work with

- Alessandro Chiesa, Madars Virza, Ian Miers, Christina Garman, Eran Tromer, Eli Ben-Sasson (*In submission)*

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 <u>arbitrary-sized arithmetic circuits</u>
  - And there are C compilers!



Pinocchio: Nearly Practical Verifiable Computation

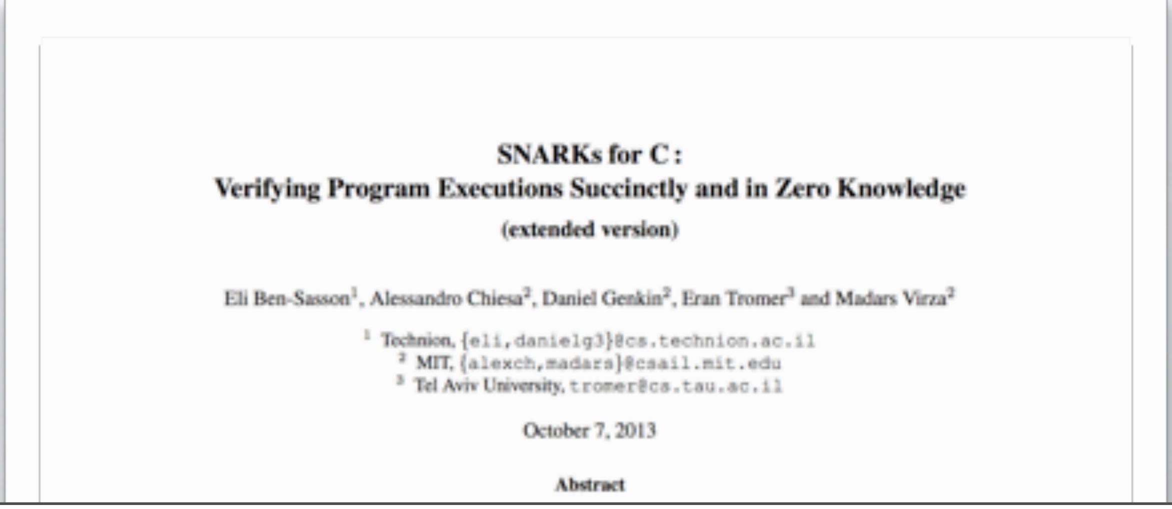
Bryan Parno
Jon Howell
Microsoft Research

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 [16–23] 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 (§5.2). 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]

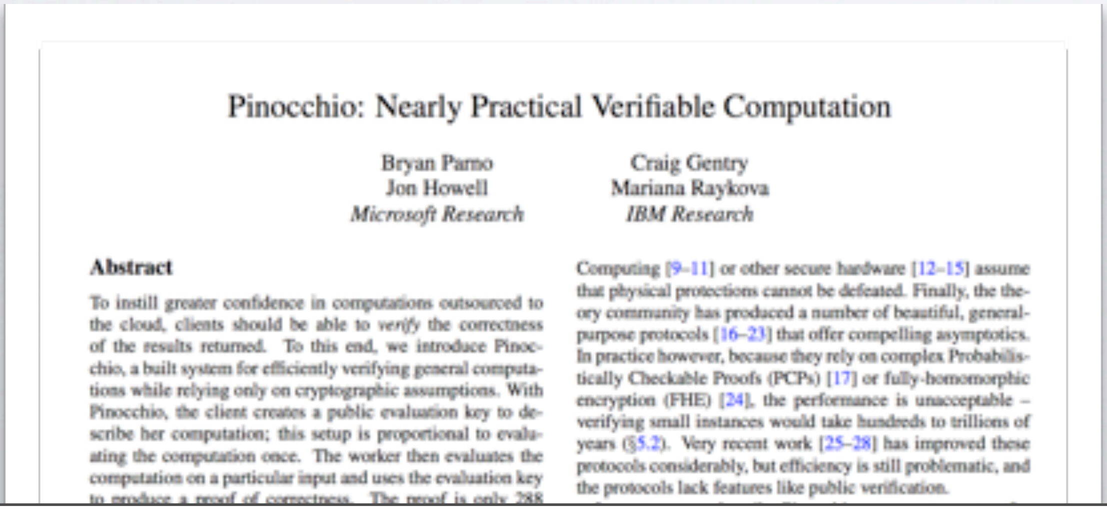
[1] Technion, {eli,danielg3}@cs.technion.ac.il
[2] MIT, {alexch,madars}@csail.mit.edu
[3] Tel Aviv University, tromer@cs.tau.ac.il

October 7, 2013

Abstract

How not 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
  - Surprise: This gives *large*, *impractical* circuits
    (proving takes a long time)



Pinocchio: Nearly Practical Verifiable Computation

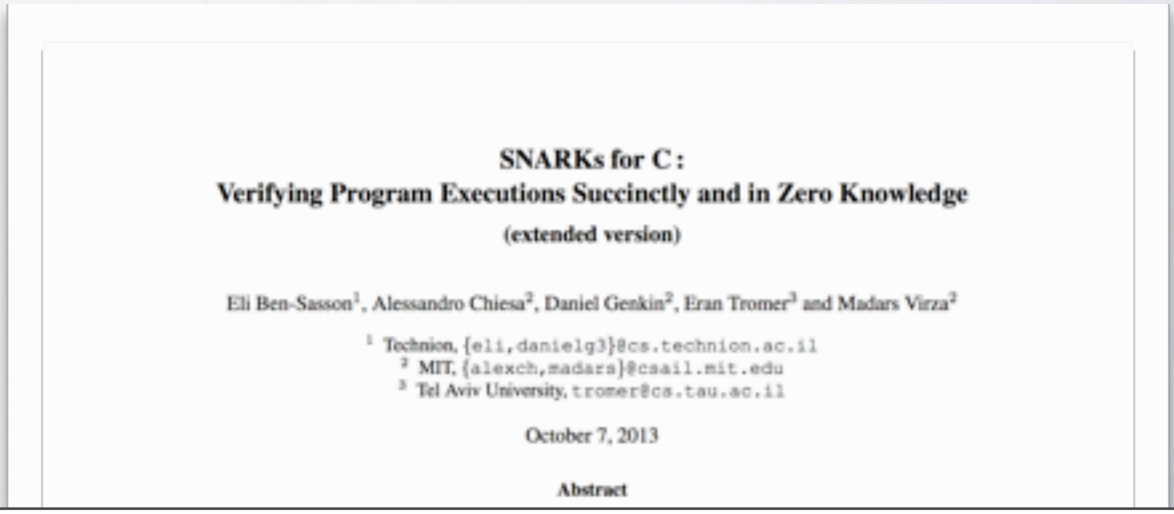
Bryan Parno
Jon Howell
Microsoft Research

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 [16–23] 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 (§5.2). 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]

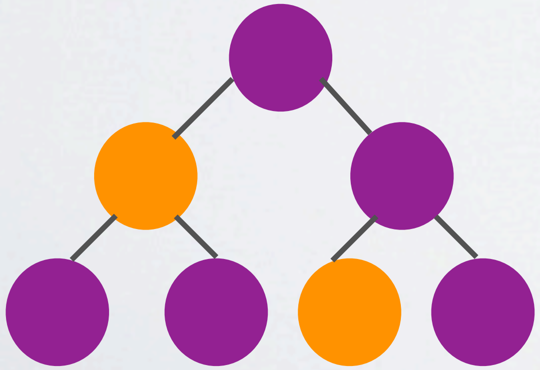
[1] Technion, {eli,danielg3}@cs.technion.ac.il
[2] MIT, {alexch,madars}@csail.mit.edu
[3] Tel Aviv University, tromer@cs.tau.ac.il

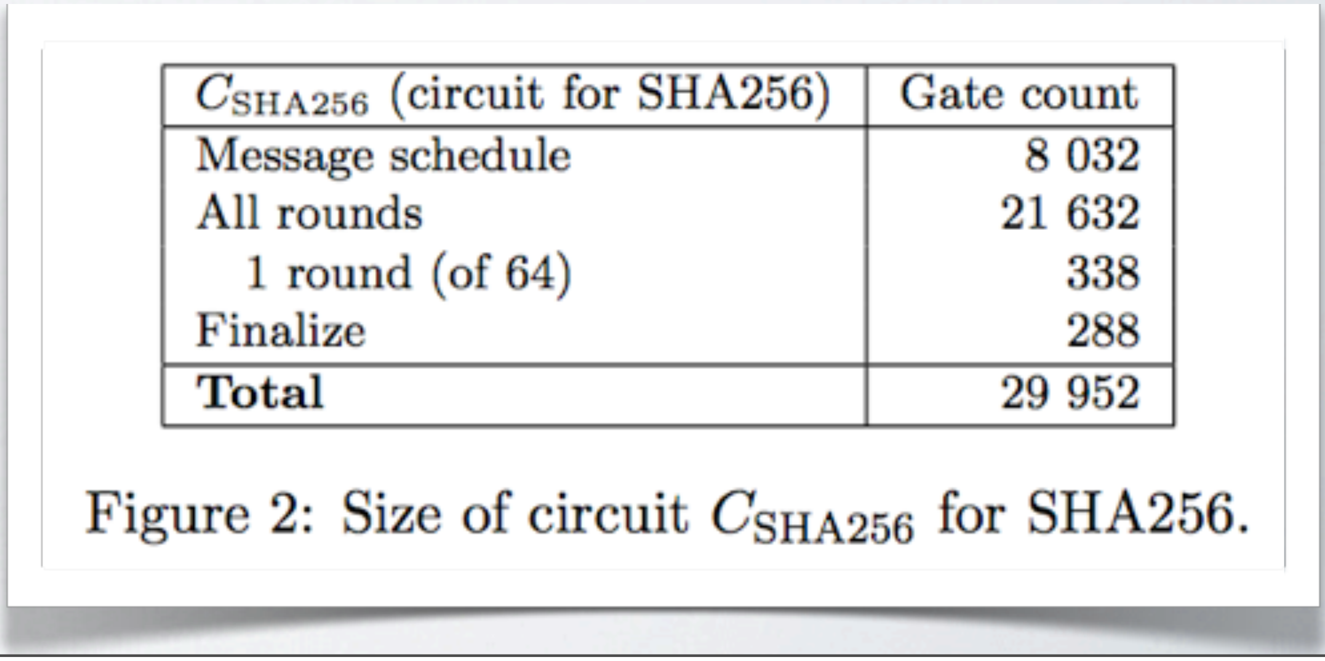
October 7, 2013

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)
  - <u>Hand-optimize everything</u>

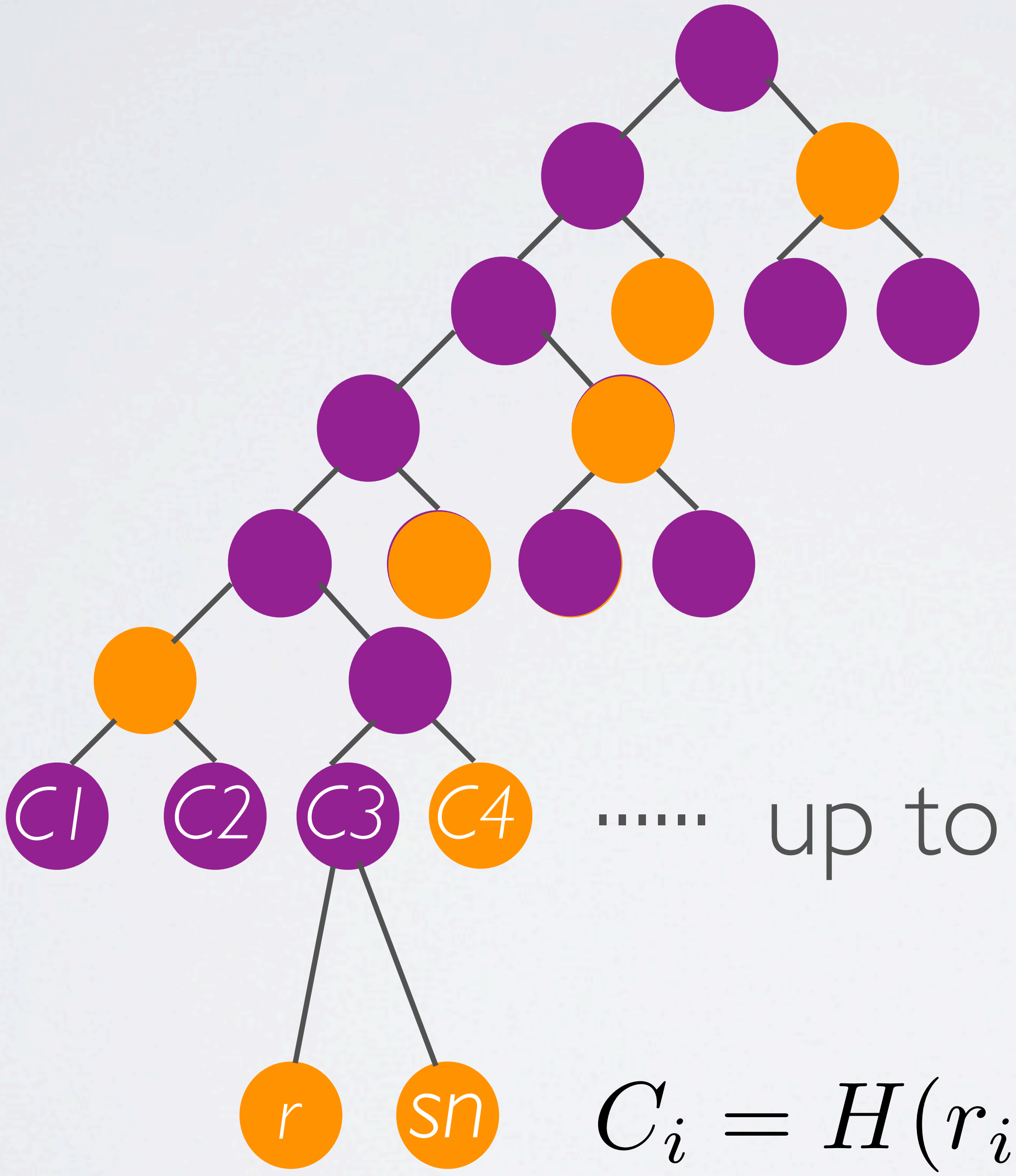




C_{SHA256} (circuit for SHA256)	Gate count
Message schedule	8 032
All rounds	21 632
1 round (of 64)	338
Finalize	288
Total	29 952

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

Proposed Zerocash tree

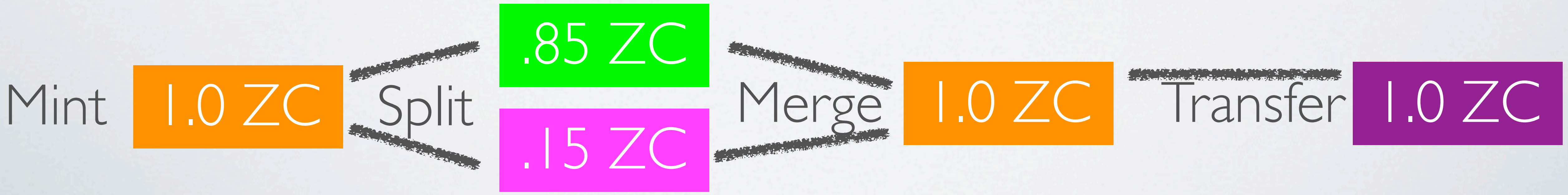


...... up to 2^{64} coins

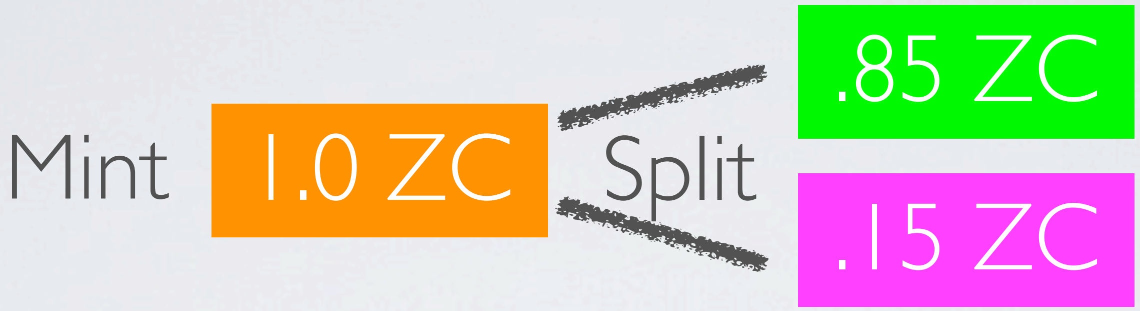
$$C_i = H(r_i || sn)$$

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

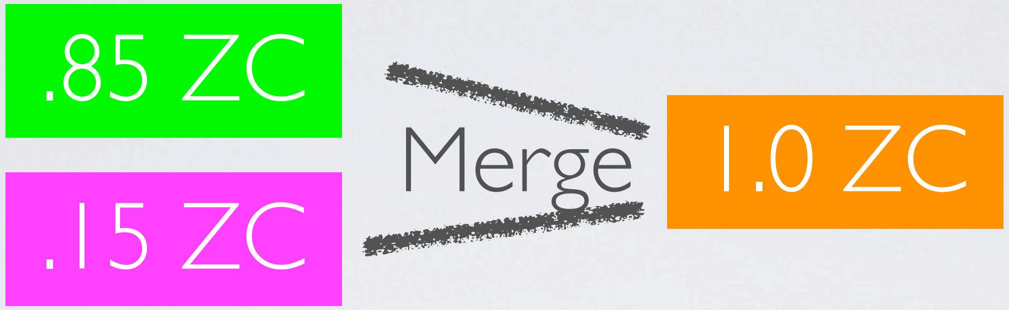


Mint 1.0 ZC Split .85 ZC .15 ZC Merge 1.0 ZC Transfer 1.0 ZC



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



.85 ZC

.15 ZC

Merge

1.0 ZC

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

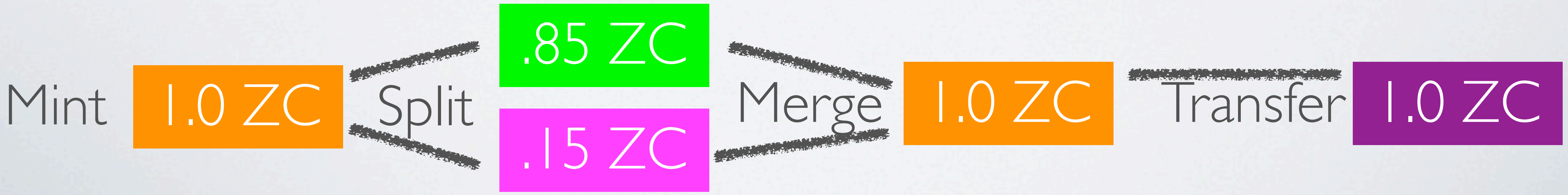
1.0 ZC Transfer 1.0 ZC

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

- 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

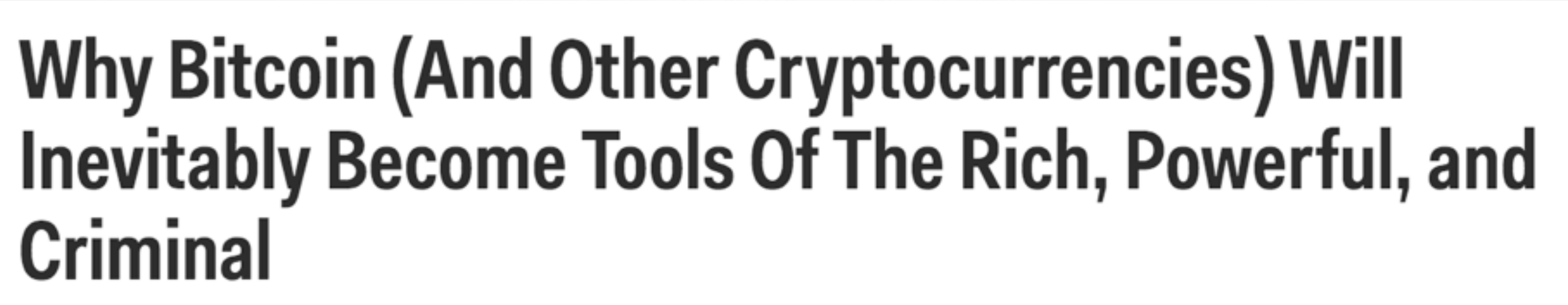
The summary

- We now have efficient and fully anonymous e-Cash
  - With practical proving times & storage costs
  - A modestly irritating set of public parameters
  - <u>And code, which we will be releasing in May</u>
  - So what next?

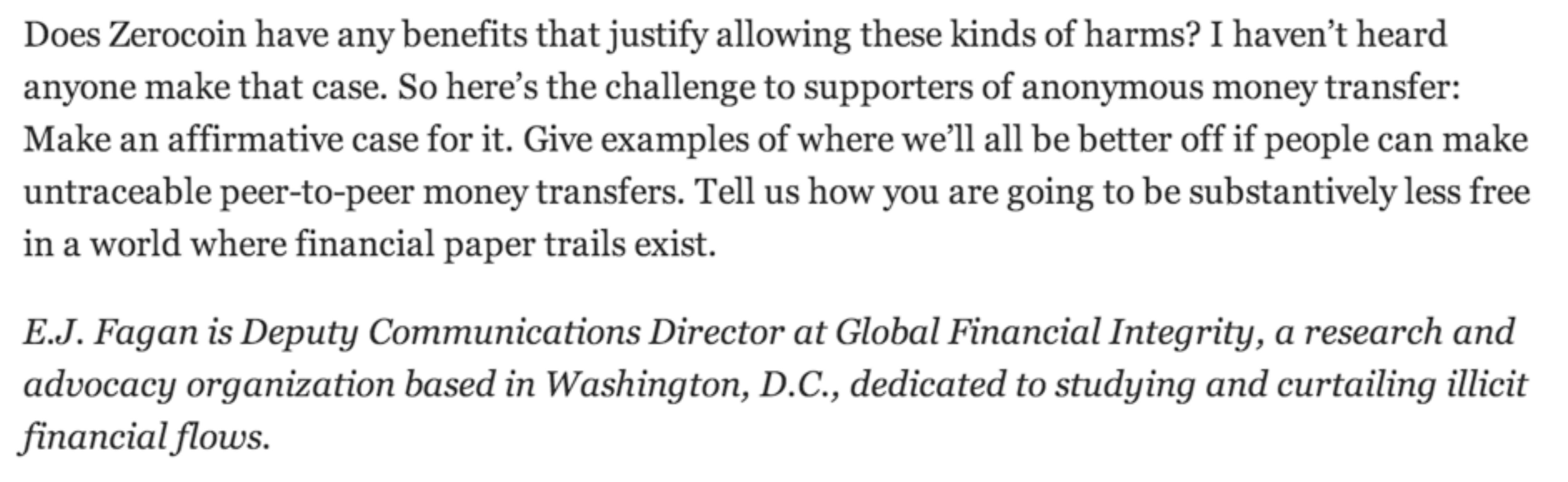
Release all the things

- This is Real World Crypto, after all
    - Our experience tells us that code is not enough
    - We need time to work out the bugs, but...
    - In May we will be launching a full alt-coin based on the Zerocoin code, mostly for testing
    - Dear god do not put real money into it

A last thought



Why Bitcoin (And Other Cryptocurrencies) Will Inevitably Become Tools Of The Rich, Powerful, and Criminal



Does Zerocoin have any benefits that justify allowing these kinds of harms? I haven't heard anyone make that case. So here's the challenge to supporters of anonymous money transfer: Make an affirmative case for it. Give examples of where we'll all be better off if people can make untraceable peer-to-peer money transfers. Tell us how you are going to be substantively less free in a world where financial paper trails exist.

E.J. Fagan is Deputy Communications Director at Global Financial Integrity, a research and advocacy organization based in Washington, D.C., dedicated to studying and curtailing illicit financial flows.