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.

<table>
<thead>
<tr>
<th>Sender</th>
<th>Recipient</th>
<th>Amount</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Starbucks</td>
<td>$8.75</td>
<td>2017.06.02 @ 10:05</td>
</tr>
<tr>
<td>Alice</td>
<td>Uber</td>
<td>$11.50</td>
<td>2017.06.02 @ 11:00</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Sender</th>
<th>Recipient</th>
<th>Amount</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>14e…</td>
<td>5b6…</td>
<td>🍀8.75</td>
<td>2017.06.02@10:05</td>
</tr>
<tr>
<td>f71…</td>
<td>88a…</td>
<td>🍀11.5</td>
<td>2017.06.02@11:00</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

"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**

addresses → 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 $\rightarrow$ rent hike)
- censorship (miners filter transactions)
If privacy is so important why isn't Bitcoin private?
Privacy vs Accountability

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?

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 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

<table>
<thead>
<tr>
<th>From</th>
<th>Enc(A)</th>
<th>From</th>
<th>Enc(S)</th>
<th>From</th>
<th>Enc(B)</th>
<th>From</th>
<th>C_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>To</td>
<td>Enc(B)</td>
<td>To</td>
<td>Enc(D)</td>
<td>To</td>
<td>Enc(E)</td>
<td>To</td>
<td>C_2</td>
</tr>
<tr>
<td>Amount</td>
<td>Enc(1)</td>
<td>Amount</td>
<td>Enc(2)</td>
<td>Amount</td>
<td>Enc(1)</td>
<td>Amount</td>
<td>C_3</td>
</tr>
<tr>
<td>Proof</td>
<td>π</td>
<td>Proof</td>
<td>π'</td>
<td>Proof</td>
<td>π&quot;</td>
<td>Proof</td>
<td>π‴</td>
</tr>
</tbody>
</table>

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 transferred 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

<table>
<thead>
<tr>
<th>From</th>
<th>Enc(A)</th>
<th>To</th>
<th>Enc(B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amount</td>
<td>Enc(1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proof</td>
<td>π</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>From</th>
<th>Enc(S)</th>
<th>To</th>
<th>Enc(D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amount</td>
<td>Enc(2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proof</td>
<td>π'</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>From</th>
<th>Enc(B)</th>
<th>To</th>
<th>Enc(E)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amount</td>
<td>Enc(1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proof</td>
<td>π''</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### 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 of knowledge**: (true statements have proofs, false ones do not)
- **NIZK**: (technical… allows using crypto in statement)
- **ZK-SNARK**: have concretely efficient constructions
  
  [libsnark.org](http://libsnark.org)
Requirements on Crypto Proof

Q2: what exactly is the statement being proved?

this requires some thought

time to have some design fun
Attempt #0: template

Transaction types

- type 1
- type 2

view of blockchain

coin
Attempt #1: plain serial numbers

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!
Attempt #2: committed serial numbers

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.

**Good:**
- cannot double spend others can't spend my coins

**Bad:**
- spend linkable to its mint
  ...
Attempt #3: ZKPoK of commitment

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 ∈ "list of prior commitments"

**well-formed**
- cm = COMM(sn;r)

- **Good:**
  - cannot double spend
  - others can't spend my coins
  - spend and mint unlinkable

- **Bad:**
  - fixed denomination
  ...
Attempt #4: variable denomination

Transaction types

**mint** \( cm, v, k, r \)

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

**spend** \( sn, v, \pi \)

Consume the value-\( v \) coin w/ serial number \( sn \).

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

- \( cm \in \) "list of prior commitments"
- \( cm = \text{COMM}(v, k; r) \) & \( k = \text{COMM}(sn; s) \)

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

**Bad:**
only hides sender
…
Attempt #5: payment addresses

Transaction types

**mint**
\[ cm_{1,v,k,r_1}, cm_{2,v,k,r_2}, cm_{3,v,k,r_3} \]

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

**spend**
\[ sn_{2,v_2,\pi_2}, sn_{1,v_1,\pi_1} \]

Consume the value-\( v \) coin w/ serial number \( sn \).

Here is a ZK proof \( \pi \) that I know secret \((cm,k,r,s,\rho,apk,ask)\) s.t.

- \( cm \in \) "list of prior commitments"
- \( cm = \text{COMM}(v,k;r) \) & \( k = \text{COMM}(apk,\rho;s) \)
- \( sn = \text{PRF}(\rho;ask) \) & \( apk = \text{PRF}(0;ask) \)

**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

Transaction types

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

- **spend**: 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,ρ^A,apk^A,ask^A) s.t. (cm^B,v^B,k^B,r^B,s^B,ρ^B,apk^B) exists

- **well-formed**: cm^A = COMM(v^A,k^A;r^A) & k^A = COMM(apk^A,ρ^A;s^A)
- **mine**: sn^A = PRF(ρ^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,ρ^B;s^B)
- **same value**: v^A = v^B

coin

- **commitment**: cm
- **serial number**: sn
- **value**: v
- **public key**: apk
- **secret key**: ρ

address

- **serial number**: sn
- **ask**: ask
- **PRF**: PRF
- **apk**: apk
- **PRF**: PRF
- **secret key**: 0

view of 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**

**Transaction types**

- **mint** \( cm_{1,v_1,k_1,r_1} \)
- **mint** \( cm_{2,v_2,k_2,r_2} \)
- **pour** \( SN_1 \) \( cm_3 \) \( SN_2 \) \( cm_4 \)
- **mint** \( cm_{5,v_5,k_5,r_5} \)
- **pour** \( SN_3 \) \( cm_6 \) \( SN_5 \) \( cm_7 \)

**view of blockchain**

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 \( \pi \) 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

**libzerocash**
Mint, Pour, VerifyTx

**arithmetic circuit for Pour NP statement**
hand optimized

**libsnark**
highly-optimized C++
ZK-SNARK library

**std crypto**
hashing, encryption, ...

- Mint
  - mint cm, v, k, r
  - 20µs
  - 70B
- Pour
  - pour sn^A, cm^C, π
  - sn^B, cm^D, ...
  - 1m
  - 1KB
- VerTx
  - acc/rej
  - 10ms

- COMM
  - Merkle Tree Path
- PRF

4 million arithmetic gates
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

• thorough analysis and vetting (even found a completeness bug! 😂)
• protocol changes
• efficiency improvements
• external security audits
• creation of clients, integration with wallets and exchanges
• generation of public parameters for the ZK-SNARK (ZK proof system)

Solar Designer
(Alexander Peslyak)
The Pain of Public Parameters

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

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. $F(x, w) = true$.

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 \( \leq n-1 \) corruptions \([\text{BCGTV15}][\text{BGG16}]\)
- 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 transferred 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...

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!

I know \( x \) s.t. \( y = F(x) \).