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
Bitcoin’s privacy problem

Bitcoin: decentralized digital 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*):

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.
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: $\log(#\text{coins})$
  – 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
  – Properties of SHA256, encryption and signature schemes

[Boneh Boyen 04] [Gennaro 04] [Groth 10]
The Zerocash scheme
Basic anonymous e-cash

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 knowledge
succinct
noninteractive argument of knowledge
zkSNARK
zkSNARK constructions for any NP statement

- **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]
    - [Ben-Sasson Chiesa Tromer Virza 14]

**SCIPR LAB**

Underlying zkSNARK used in Zerocash
zkSNARK

with great power comes great functionality

\( \text{cm} \)  
(coin commitment)

\( \text{commit} \)

\( r \)

\( \text{sn} \)  
(serial number)
I hereby spend $v$ BTC to create $cm$, and here is $k, r'$ to prove consistency.

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

Adding variable denomination

Minting:

Spending:
Adding direct anonymous payments

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

Minting, spending analogous to above.

Sending?

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

1. Create coin using $a_{pk}$ of payee.
2. Send coin secrets ($v, \rho, r', r''$) to payee out of band, or encrypted to payee’s public key.
Pouring Zerocash coins

Single transaction type capturing:

Sending payments
Making change
Exchanging into bitcoins
Transaction fees

Pour old Zerocash coin

\( v_1 \)

\( v_2 \)

\( \text{dest}_1 \)

\( \text{dest}_2 \)

\( v_{\text{pub}} \)

old Zerocash coin

new Zerocash coin

value \( v_1 \) to \( \text{dest}_1 \)

new Zerocash coin

value \( v_2 \) to \( \text{dest}_2 \)

public bitcoins

of value \( v_{\text{pub}} \)

\( \text{sn}_1 \)

\( \text{sn}_2 \)

\( \text{cm}_1 \)

\( \text{cm}_2 \)

proof

the old coins were \textbf{valid}, and
values of old coins = \( v_1 + v_2 + v_{\text{pub}} \)
Pouring Zerocash coins

Single transaction type capturing:
- Sending payments
- Making change
- Exchanging into bitcoins
- Transaction fees

Pour

old Zerocash coin

old Zerocash coin

\( v_1 \)
\( v_2 \)
\( dest_1 \)
\( dest_2 \)
\( v_{\text{pub}} \)

new Zerocash coin
value \( v_1 \) to \( dest_1 \)
new Zerocash coin
value \( v_2 \) to \( dest_2 \)
public bitcoins
of value \( v_{\text{pub}} \)

\( sn_1 \)
\( sn_2 \)
\( cm_1 \)
\( cm_2 \) ...

\[ \text{proof} \]

the old coins were valid, and values of old coins = \( v_1 + v_2 + v_{\text{pub}} \)
# Example of a Zerocash Pour transaction

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td>1c4ac4a110e863deeca050dc5e5153f2b7010af9</td>
</tr>
<tr>
<td>sn_1</td>
<td>a365e7006565f14342df9096b46cc7f1d2b9949367180fdd8ede4090eee30b7dc</td>
</tr>
<tr>
<td>sn_2</td>
<td>6937031dce13facdebe79e8e2712ffad2e980c9114ece8ca9b25fc88d7b3b52</td>
</tr>
<tr>
<td>cm_1</td>
<td>a4d015440f9cfce0uc3ca3a38cf04058262d74b60cb14ec6d6e3e047694580103</td>
</tr>
<tr>
<td>cm_2</td>
<td>2ca1f833b63ac827ba6ae69b53edc855e66e2c2da24dfb6d5b04fa5d042dc772</td>
</tr>
<tr>
<td>v_pub</td>
<td>00000000000000000000042</td>
</tr>
<tr>
<td>pubkeyHash info</td>
<td>8f9a43f0fe28bef052ec209724bb0e502fbb5427</td>
</tr>
<tr>
<td>SigPK</td>
<td>2dd489d97daa8c006cb4049ee1699b16a16d108d43</td>
</tr>
<tr>
<td>Sig</td>
<td>f1d2df294e6986ac86df7b36c94bcd32bee15a3835982f32db3342cb4bedcb78ce116bac69e</td>
</tr>
<tr>
<td>MAC_1</td>
<td>b8a5917ea15387a9670bc9e3ec5e395240ce1b0f7002760ca92d1835cb7f629</td>
</tr>
<tr>
<td>MAC_2</td>
<td>ade6218b3a17d609936ec68947b2bb446f12698d4bcafa85fcbf39fb546603a</td>
</tr>
<tr>
<td>ciphertext_1</td>
<td>048070fe125bda9f3ae6a7c08b65adb2a438468d7243c74e80abc5b74de3524a987a2e3ed075d54ae7a53866973ea50704e08954f5d80caea214ce572f42dc6676f0e59d5b1ed68ad33b0c73cf9eac671d8f0126d86b667b319d255d7002d0a02d82efc47fd8fd648057fa823a25dd35f52e866ed65cee229db568166e46967baf4d2303af7fe09d24b8e30277336cb7d8c81d3c786f1547fe0d00c29b63bd9272aad873f1a2b667fa575e</td>
</tr>
<tr>
<td>ciphertext_2</td>
<td>0493110814391b0b5cab99a9225062354987c8b8f60496985ca527a770555b4979a50099efc5a359bdf0411983388f5ade840a0d408116f3d9f38641d21798aaf9817f6420caf19a2dc187c9abc14bd978624e80a2cc27063eb67f8b78c42ee01edbfccddbeb06e0a5b6aeeac6cb017069c8ebe8eaa82fa5e0f7680a4e2466d72bc3423e873820b2ed4eb954e9216b566cc140de79351abf47254d122a35f17f840156b7b1fe9492792adc</td>
</tr>
<tr>
<td>zkSNARKproof</td>
<td>a43c36d6e02e5c15dc8a37ebc51885cf865cda044b1c22cb3ed97b7787f788daceb240c40a0cc3f825843efdf1aafdeccf532bc5afafeef9e9d3975726f2ca8292286a8dd48f8d2ab13b98c6fa1caaf2a38b2544855b17c4ee7a0c9e57592e1ed233d4a3e276bb9de5ba365947896f17000209f7508bdf611e206c2087618c58208e3658cc00188431f53885139d0180a3e6d89432287699e76e7d832a5f5d3cd30874ff0982d965b8e7c02649e3df29f58d0c5dac75637412980251806dfe9ce5fc8c40dbfe87dab1146799e6154fb9623d3fbae97c78ad1708b17992715df4d31c9451e0b59df7dc506ddad84ae98475d4be530eb501925dfd72981a2970a3799523b99a8e50dd0eaba5306c1be05</td>
</tr>
</tbody>
</table>

~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

libsark
zkSNARK
SCIPR LAB

Performance (quadcore desktop)

<table>
<thead>
<tr>
<th></th>
<th>bitcoind</th>
</tr>
</thead>
<tbody>
<tr>
<td>Libsark</td>
<td>Instantiate Zerocash</td>
</tr>
<tr>
<td></td>
<td>primitives and parameters</td>
</tr>
<tr>
<td>SCIPR Lab</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Setup</th>
<th>Mint</th>
<th>Pour</th>
<th>Verify Transaction</th>
<th>Receive</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&lt;2 min, 896MB params</td>
<td>23 μs</td>
<td>46 s, 1KB transaction</td>
<td>&lt;9 ms/transaction</td>
<td>&lt;2 ms/transaction</td>
</tr>
</tbody>
</table>
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 $r', r'', \rho, a_{pk}$ 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.