Bitcoin review

**Bitcoin idea:** Decentralize digital currency

**Double spending:** the same single digital token can be spent more than once
Bitcoin idea: Decentralize digital currency
Double spending: the same single digital token can be spent more than once
The blockchain

It is an open, distributed ledger that can record transactions between two parties efficiently and in a verifiable and permanent way.

- Sender
- Receiver
- Amount
- Others

Everything is totally public!!
Privacy

Privacy and Fungibility

Satoshi nakamoto
Privacy section of
paper of '09

F. Reid and H. Martin, “An analysis of anonymity in the Bitcoin
system,” in SocialCom/PASSAT ’11
ZeroCash: Decentralized Anonymous Payments from Bitcoin
Privacy

Privacy and Fungibility

Satoshi Nakamoto
Privacy section of paper of '09

Privacy solutions

Mix (tambler or Laundries): A mix allows users to entrust a set of coins to a pool operated by a central party and then, after some interval, retrieve different coins from the pool.

- The delay to reclaim coin must be large
- The mix can still trace coins
- The mix can shutdown everything and run

When the cookie meets the blockchain: Privacy risks of web payments via cryptocurrencies
Zerocoin

- No third part
- Strong privacy guarantees
- No digital signature
• No third part
• No digital signature
Strong privacy guarantees
Zerocoin

MINT
Mint a coin from the public ledger into a private coin

MINT
The owner of the coin does not need to be revealed

Repeat
Minting and spending can be repeated
MINT

Mint a coin from the public ledger into a private coin
MINT

The owner of the coin does not need to be revealed
Repeat

Minting and spending can be repeated
Why Zerocoin does not work
Why Zerocoin does not work

• More than a full-fledged electronic currency, it looks like a cool “decentralized mix”
Why Zerocoin does not work

- More than a full-fledged electronic currency, it looks like a cool “decentralized mix”

- Its performance are not competitive with Bitcoin (e.g. Zerocoin spend transactions are also comparatively computationally intensive to verify taking about half a second to do so).
Why Zerocoin does not work

- More than a full-fledged electronic currency, it looks like a cool “decentralized mix”

- Its performance are not competitive with Bitcoin (e.g. Zerocoin spend transactions are also comparatively computationally intensive to verify taking about half a second to do so).

- It lacks critical features required of full-fledged anonymous payments (e.g., fixed denominations, still possible some graph analysis).

- Also incorrect use or predictable use of Zerocoin mint and spend transactions such as always minting and spending at regular intervals or using the same IP address for a mint and spend can possibly compromise anonymity thus some sort of required...
ZeroCash: Decentralized Anonymous Payments from Bitcoin
Zero - Knowledge Proof
Zero - Knowledge Proof
Zero - Knowledge Proof
Zero - Knowledge Proof
zk-Snark
More formally

Accountant
More formally
More formally

Prover

NP algorithm

NP statement

Proof

Verifier
More formally

NP-statement properties

Prover

NP statement

Verifier

NP algorithm

Proof
More formally

NP-statement properties
- Succint

Prover

NP statement

Proof

Verifier

NP algorithm
More formally

NP-statement properties
- Succint
- Non interactive

Prover

NP statement

Verifier

NP algorithm

Proof
More formally:

NP-statement properties:
- Succint
- Non interactive
- Argument

Prover → NP statement → Proof → Verifier
More Formally

NP-statement properties
- Succint
- Non interactive
- Argument
- Knowledge

Prover

NP statement

Answer

Verifier

NP algorithm

Proof
More Formally

NP-statement properties
- Succint
- Non interactive
- Argument
- Knowledge

Prover

zk-Snark
Verifier

NP algorithm

NP statement
Proof
Decentralized Anonymous Payment

- Setup
- Mint
- Pour
- Receive
- Create Addresses
- Verify Transaction
Setup

- Executed by a trusted party

- Public parameters are available
Trusted Setup

- Ran once and intermediate results erased
- No further trust
- Risk of forgery if compromised
Create Address

public parameters → Create Address → address key pair (a_pk, a_sk)
Minting operation

- Obtained via SHA256
- Computationally binding
- Statistically hiding

explain why the sn
Minting operation

- Obtained via SHA256
- Computationally binding
- Statistically hiding

Serial number (sn)

explain why the sn
Minting operation

- Obtained via SHA256

- Computationally binding
- Statistically hiding

Serial number (sn)

R

Commitment Scheme

Coin commitment (cm)

Explain why the sn
Minting operation

```
R

Serial number(sn)

Commitment Scheme
- Obtained via SHA256
- computationally binding
- statistically hiding

Coin commitment(cm)
```

explain why the sn

```
zepto := (R, sn, cm)
```
Minting operation

- Obtained via SHA256
- Computationally binding
- Statistically hiding

Serial number (sn) → Commitment Scheme → Coin commitment (cm)

explain why the sn

\[ (R, sn, cm) \rightarrow T\text{-MINT cm} \]
Minting operation

Serial number (sn)

- Obtained via SHA256
- Computationally binding
- Statistically hiding

```
:= (R, sn, cm)
```

T-MINT

Coin commitment (cm)

Escrow Pool
Spending Transaction

\[
\text{cm(\hspace{0.5em}) cm(\hspace{0.5em}) cm(\hspace{0.5em}) cm(\hspace{0.5em}) cm(\hspace{0.5em})}
\]
Spending Transaction

Coin Commitment list: cm(денг) cm(денг) cm(денг) cm(денг) cm(денг)
Spending Transaction

Coin Commitment list :cm(_coin_commitment) cm(coin_commitment) cm(coin_commitment) cm(coin_commitment) cm(coin_commitment)

Reveal the serial number

31.3000.1440474
Reveal the serial number

31.3000.1440474
Reveal the serial number

31.3000.1440474
Publish a zk-snark proof

I know r such that COMM_r(sn) appears in the list CMList of coin commitments”
Improving performance

- Define a collision resistant function CRH
- Avoid explicit CMList representation
Improving performance

- Define a collision resistant function CRH
- Avoid explicit CMLlist representation
Improving performance

- Define a collision resistant function CRH
- Avoid explicit CMList representation

“I know r such that COMMr(sn) appears as a leaf in a CRH-based Merkle tree whose root is rt”
Minting - general value

R

serial number

K := CHM(R, sn)

Commitment Scheme

Value

General Value Payment

Commitment

Coin commitment
Minting - general value

General Value Payment

R

serial number

Commitment Scheme

K := CHM(R, sn)

Coin commitment

Value

I am using up a coin with value v unique(sn) and I know R, s that are consistent with cm
Minting - general value

R

Commitment Scheme

K := CHM(R, sn)

Coin commitment

s

Value

General Value Payment

I am using up a coin with value v unique(sn) and I know R, s that are consistent with cm
DAP construction - sending anonymous payments

a_pk

Commitment Scheme

K := CHM(R,sn)

Coin commitment

Value

General Value Payment

PRF

Pseudorandom Function

Serial number

I am using up a coin with value v unique(sn) and I know R, s, ro, a_pk that are consistent with cm
DAP construction - sending anonymous payments

K := CHM(R, sn)

Commitment Scheme

Coin commitment

Value

General Value Payment

PRF

Pseudorandom Function

I am using up a coin with value v unique(sn) and I know R, s, ro, a_pk that are consistent with cm
Minting transaction

\[
\text{T-MINT}
\]

\[
\begin{align*}
\text{value} \\
s \\
k &= \text{CHM}(a_{\text{pk}}, ro) \\
\text{cm}
\end{align*}
\]
Pouring transaction

- subdivide coins into smaller denominations
- merge coins
- transfer ownership of anonymous coins
- or make public payments
Pouring transaction
Pouring transaction

V1 V2 DEST1 DEST2 VPUB

- public parameters
- Merkle tree root
- old coins c1 and c2
- Old addresses
- authentication path from cm(c1) to the root
Pouring transaction

V1 V2 DEST1 DEST2 VPUB

public parameters
Merkle tree root
old coins c1 and c2
Old addresses
authentication path from cm(c1) to the root

new zcash coin
new zcash coin
public bitcoins
Pouring transaction

- public parameters
- Merkle tree root
- old coins c1 and c2
- Old addresses
- authentication path from cm(c1) to the root

V1 V2 DEST1 DEST2 VPUB

new zcash coin
new zcash coin
public bitcoins

sn1 sn2 cm1 cm2 vpub
Proving the pouring statement

Given the Merkle-tree root \( rt \), serial number snold, and coin commitments \( cmnew \), \( cmnew \), I know coinscold, \( cnew \), \( cnew \), and address secret key old such that:
- coins are well formed
- The address secret key matches the public key
- The serial number is computed correctly
- The coin commitment appears as a leaf of the MT
- balance is preserved
Proving the pouring statement
<table>
<thead>
<tr>
<th><code>root</code></th>
<th>1c4a4c3110e863deeca050dc5e533f2b7010af9</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sn_1</code></td>
<td>a365e7088655f1393212930c66c7f1d2b92f953c7136d8fdd3c4090e3c39bf0c</td>
</tr>
<tr>
<td><code>sn_2</code></td>
<td>69370313c5e13f9ace2be79e82712f3ad2e9800c51d404c38c925b2f68cf73b52</td>
</tr>
<tr>
<td><code>cm_1</code></td>
<td>a4d010440f9caee0c3c5a38c00205826276d7460cb14ced63e25d9b2e0103</td>
</tr>
<tr>
<td><code>cm_2</code></td>
<td>2c8163b6b1ac98724a6e6b515ed5e62c22da2419e3e3045a503d2c2772</td>
</tr>
<tr>
<td><code>v_pub</code></td>
<td>00000000000000</td>
</tr>
<tr>
<td><code>pubkeyHash</code></td>
<td>info</td>
</tr>
<tr>
<td><code>SigPK</code></td>
<td>2dd489d97ca18eb06c6b04961e9a6fd180841</td>
</tr>
<tr>
<td><code>Sig</code></td>
<td>f1d2d2f124e99e6c83f6d7b3e6c93b6f321b2c35a38359c82532db33422cb4fbedb78ce16eac98e</td>
</tr>
<tr>
<td><code>MAC_1</code></td>
<td>b8a5d17ea1e087e9c0700b09e3ce6f352048e8e7e782f7e0a02a3d1e35db74629</td>
</tr>
<tr>
<td><code>MAC_2</code></td>
<td>ade2613b1317b61c9336a6c6b84d7b3b2b4464f1269840bcafa85f5cbf30f954663a</td>
</tr>
<tr>
<td><code>ciphertext_1</code></td>
<td>048070f6125d6af91ae67a7c0866ab0b2a348468d724374e78a9a0b574dc987a2936c52e076554ae7a53866973eaa5af70c80d5872</td>
</tr>
<tr>
<td></td>
<td>4f9d6d80c9eae2174e57242f3667760e6f5d51ed86a333b6c73c3f93ece671d8f010212e6667b5139d255701220a2d82e9c77df8fda6f8</td>
</tr>
<tr>
<td></td>
<td>057a9832a35d5f2f86e6d6e2279d56164e6466765bf4f8b63438f7fe0924be30377336cb7dce81d3c786f1547fe0d0c592b63c0d972caad8f7b3a2b6d67f5a7756</td>
</tr>
<tr>
<td><code>ciphertext_2</code></td>
<td>049311c0813a1b2b3b3c5a92123635487d5c8d60495993a52c37a70132b697a5050999c9c3a13b9e384141983188f1b33d5904a0d0</td>
</tr>
<tr>
<td></td>
<td>6819f61b6c884df4d17198a6f8176e420eaf12a2dc18c79acbf4b9c78624579ac272b563a666f74b8c2ae17edf0b4ddeb5069ba586</td>
</tr>
<tr>
<td></td>
<td>e4acc6c6b179df8e82e08e8eaf5a5f0e6780a4e266d212cc3243e873102b2d2e9b95e9216b566c10de99515bf41234d12a35f</td>
</tr>
<tr>
<td><code>zkSNARKproof</code></td>
<td>2f8b01514f5be7b1f9eb79427204da</td>
</tr>
</tbody>
</table>

70.
Verify Transaction

- Mint and pour must be verified
- Verified by nodes and users
Receive coins

- Scan the ledger
- For each payment to the public address earn the corresponding amount of money
Privacy

**Ledger indistinguishability:** Nothing revealed beside public information

**Balance:** Can't own more money than received or minted

**Transaction non-malleability:** Can't manipulate transactions en route to ledger
ZeroCash Performance
# ZeroCash Performance

<table>
<thead>
<tr>
<th>Setup</th>
<th>&lt; 2 min (896MB params)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Zerocash Performance

<table>
<thead>
<tr>
<th>Setup</th>
<th>&lt; 2 min (896MB params)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mint</td>
<td>&lt; 23us (72B transaction)</td>
</tr>
</tbody>
</table>
# ZeroCash Performance

<table>
<thead>
<tr>
<th></th>
<th>Setup</th>
<th>Mint</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time</strong></td>
<td>&lt; 2 min (896 MB params)</td>
<td>&lt; 23 us (728 bytes transaction)</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Zerocash Performance

<table>
<thead>
<tr>
<th></th>
<th>Time</th>
<th>Data Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setup</td>
<td>&lt; 2 min</td>
<td>896MB params</td>
</tr>
<tr>
<td>Mint</td>
<td>&lt; 23us</td>
<td>72B transaction</td>
</tr>
<tr>
<td>Pour</td>
<td>&lt; 46s</td>
<td>1KB transaction</td>
</tr>
</tbody>
</table>
# Zerocash Performance

<table>
<thead>
<tr>
<th>Action</th>
<th>Time/Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setup</td>
<td>&lt; 2 min (896MB params)</td>
</tr>
<tr>
<td>Mint</td>
<td>&lt; 23us (72B transaction)</td>
</tr>
<tr>
<td>Pour</td>
<td>&lt; 46s (1KB transaction)</td>
</tr>
<tr>
<td>Verify Transactio</td>
<td>&lt; 9 ms/transaction</td>
</tr>
</tbody>
</table>
# Zerocash Performance

<table>
<thead>
<tr>
<th>Action</th>
<th>Time (Params/Transaction)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setup</td>
<td>&lt; 2 min (896MB params)</td>
</tr>
<tr>
<td>Mint</td>
<td>&lt; 23us (72B transaction)</td>
</tr>
<tr>
<td>Pour</td>
<td>&lt; 46s (1KB transaction)</td>
</tr>
<tr>
<td>Verify Transaction</td>
<td>&lt; 9 ms/transaction</td>
</tr>
<tr>
<td>Receive</td>
<td>&lt; 2 ms/transaction</td>
</tr>
</tbody>
</table>
Zerocash: Decentralized Anonymous Payments from Bitcoin
Zerocash flaws

- Brand new cryptography
- Impossibility to verify forgery
- Poison pill attack (paper extended version)
- Trusted setup
- The fact that it is a US based company does not help confidence in the currency
Summary
Summary
Summary
Summary
Summary
Summary
Zerocash:
Decentralized
Anonymous Payments
from Bitcoin