Zerocash Explained

Abhiram Kothapalli

Carnegie Mellon University

The Zerocash Protocol

The Zerocash Protocol

- Zerocash was introduced by [SCG+14]

Carnegie Mellon University

The Zerocash Protocol

- Zerocash was introduced by [SCG+14]
- Proposed an anonymous digital currency that hides both transaction participants and value.

The Zerocash Protocol

- Zerocash was introduced by [SCG$^+$14]
- Proposed an anonymous digital currency that hides both transaction participants and value.
- It's core technology is the Pinocchio zero-knowledge proof system [PHGR13]

Carnegie Mellon University

The Zerocash Protocol

- Zerocash was introduced by [SCG+14]
- Proposed an anonymous digital currency that hides both transaction participants and value.
- It's core technology is the Pinocchio zero-knowledge proof system [PHGR13]
- Zcash is the corresponding commercial realization that is now worth $2.08B.

Outline

1. The Decentralized Anonymous Payment Scheme Functionality
2. Constructing a Decentralized Anonymous Payment Scheme
3. Zerocash in the Wild

Carnegie Mellon University

The DAPS Functionality

A *decentralized anonymous payment scheme* is defined by three soundness properties.

Carnegie Mellon University

The DAPS Functionality

A *decentralized anonymous payment scheme* is defined by three soundness properties.

- **Ledger Indistinguishability:** The ledger does not reveal transaction amounts and transaction participants.

The DAPS Functionality

A *decentralized anonymous payment scheme* is defined by three soundness properties.

- **Ledger Indistinguishability:** The ledger does not reveal transaction amounts and transaction participants.
- **Transaction Non-Malleability:** No adversary can modify a valid transaction.

Carnegie Mellon University

The DAPS Functionality

A *decentralized anonymous payment scheme* is defined by three soundness properties.

- **Ledger Indistinguishability:** The ledger does not reveal transaction amounts and transaction participants.
- **Transaction Non-Malleability:** No adversary can modify a valid transaction.
- **Balance:** No adversary can own more money than minted or recieved via payment.

Carnegie Mellon University

The Strawman Construction

Baseline System:

- Assume a blockchain maintaining BTC transactions.
- **Minting:** Add a mechanic to lift 1 BTC into 1 ZEC.
- **Spending:** Add a mechanic to lower 1 ZEC into 1 BTC while hiding origin.

The Strawman Construction: Minting

Suppose a user U wants to mint 1 ZEC.

The Strawman Construction: Minting

Suppose a user U wants to mint 1 ZEC.

- U pays 1 BTC to a backing escrow pool.

The Strawman Construction: Minting

Suppose a user U wants to mint 1 ZEC.

- U pays 1 BTC to a backing escrow pool.
- U samples serial number sn, randomness r and computes

$$\text{Commitment cm} \leftarrow \text{com}(\text{sn}; r)$$
$$\text{Private Coin } \mathbf{c} \leftarrow (r, \text{sn}, \text{cm})$$

Carnegie Mellon University

The Strawman Construction: Minting

Suppose a user U wants to mint 1 ZEC.

- U pays 1 BTC to a backing escrow pool.
- U samples serial number sn, randomness r and computes

$$\text{Commitment cm} \leftarrow \text{com}(\text{sn}; r)$$
$$\text{Private Coin } \mathbf{c} \leftarrow (r, \text{sn}, \text{cm})$$

- U broadcasts a mint transaction $\text{tx}_{\text{Mint}} = \text{cm}$ to the BTC blockchain.

The Strawman Construction: Minting

Suppose a user U wants to mint 1 ZEC.

- U pays 1 BTC to a backing escrow pool.
- U samples serial number sn, randomness r and computes

$$\text{Commitment cm} \leftarrow \text{com}(\text{sn}; r)$$
$$\text{Private Coin } \mathbf{c} \leftarrow (r, \text{sn}, \text{cm})$$

- U broadcasts a mint transaction $\text{tx}_{\text{Mint}} = \text{cm}$ to the BTC blockchain.
- If U has paid 1 BTC to escrow, BTC miners set

$$\text{CMLIST} = \text{CMLIST} \| \text{cm}$$

Carnegie Mellon University

The Strawman Construction: Spending

Suppose another user V wants to spend 1 private coin $\mathbf{c} := (r, \mathsf{sn}, \mathsf{cm})$.

Carnegie Mellon University

The Strawman Construction: Spending

Suppose another user V wants to spend 1 private coin $\mathbf{c} := (r, \mathrm{sn}, \mathrm{cm})$.

- V writes a zkSNARK proof π asserting the following strawman statement

Strawman Statement

For **public** $(\mathrm{sn}, \mathrm{CMLIST})$,
I know **private** r,
such that $\mathrm{com}(sn, r) \in \mathrm{CMLIST}$.

Carnegie Mellon University

The Strawman Construction: Spending

Suppose another user V wants to spend 1 private coin $\mathbf{c} := (r, \mathrm{sn}, \mathrm{cm})$.

- V writes a zkSNARK proof π asserting the following strawman statement

Strawman Statement

For **public** $(\mathrm{sn}, \mathrm{CMLIST})$,
I know **private** r,
such that $\mathrm{com}(sn, r) \in \mathrm{CMLIST}$.

- V broadcasts a spend transaction $\mathrm{tx}_{\mathrm{Spend}} = (\mathrm{sn}, \pi)$.

The Strawman Construction: Spending

Suppose another user V wants to spend 1 private coin $\mathbf{c} := (r, \mathrm{sn}, \mathrm{cm})$.

- V writes a zkSNARK proof π asserting the following strawman statement

Strawman Statement

For **public** $(\mathrm{sn}, \mathrm{CMLIST})$,
I know **private** r,
such that $\mathrm{com}(sn, r) \in \mathrm{CMLIST}$.

- V broadcasts a spend transaction $\mathrm{tx}_{\mathrm{Spend}} = (\mathrm{sn}, \pi)$.
- BTC miners award V 1 BTC if π is valid and sn is not in a prior spend transaction.

The Strawman Construction: Spending

Suppose another user V wants to spend 1 private coin $\mathbf{c} := (r, \text{sn}, \text{cm})$.

- V writes a zkSNARK proof π asserting the following strawman statement

Strawman Statement

For **public** (sn, CMLIST),
I know **private** r,
such that $\text{com}(sn, r) \in \text{CMLIST}$.

- V broadcasts a spend transaction $\text{tx}_{\text{Spend}} = (\text{sn}, \pi)$.
- BTC miners award V 1 BTC if π is valid and sn is not in a prior spend transaction.

Anonymity holds because r is not revealed and therefore tx_{Spend} is not tied to cm.

Version II: Achieving a Sublinear Statement

Problem: With public input $(\mathsf{sn}, \mathsf{CMLIST})$, proof generation and verification time grows linearly with $|\mathsf{CMLIST}|$.

Version II: Achieving a Sublinear Statement

Problem: With public input $(\mathsf{sn}, \mathsf{CMLIST})$, proof generation and verification time grows linearly with $|\mathsf{CMLIST}|$.

Solution: Store CMLIST in a Merkle tree and only make the root a part of the statement.

Version II: Achieving a Sublinear Statement

Problem: With public input $(\mathsf{sn}, \mathsf{CMLIST})$, proof generation and verification time grows linearly with $|\mathsf{CMLIST}|$.

Solution: Store CMLIST in a Merkle tree and only make the root a part of the statement.

Version II Statement

Carnegie Mellon University

Version II: Achieving a Sublinear Statement

Problem: With public input $(\mathsf{sn}, \mathsf{CMLIST})$, proof generation and verification time grows linearly with $|\mathsf{CMLIST}|$.

Solution: Store CMLIST in a Merkle tree and only make the root a part of the statement.

Version II Statement

For **public** $(\mathsf{sn}, \mathsf{rt})$,

Version II: Achieving a Sublinear Statement

Problem: With public input $(\mathsf{sn}, \mathsf{CMLIST})$, proof generation and verification time grows linearly with $|\mathsf{CMLIST}|$.

Solution: Store CMLIST in a Merkle tree and only make the root a part of the statement.

Version II Statement

For **public** $(\mathsf{sn}, \mathsf{rt})$,
I know **private** (r, π_{mk}),

Carnegie Mellon University

Version II: Achieving a Sublinear Statement

Problem: With public input $(\text{sn}, \text{CMLIST})$, proof generation and verification time grows linearly with $|\text{CMLIST}|$.

Solution: Store CMLIST in a Merkle tree and only make the root a part of the statement.

Version II Statement

For **public** (sn, rt),
I know **private** (r, π_{mk}),
such that Merkle proof π_{mk} attests that $\text{com}(sn, r) \in \text{Tree}(\text{CMLIST})$.

Carnegie Mellon University

Version III: Transferring Spending Rights

Problem: Suppose user U creates a coin $\mathbf{c}$ and sends it to V. How do we ensure that U can no longer spend $\mathbf{c}$.

Version III: Transferring Spending Rights

Problem: Suppose user U creates a coin **c** and sends it to V. How do we ensure that U can no longer spend **c**.

Solution: Introduce ephemeral public-private address pairs.

Version III: Transferring Spending Rights

Problem: Suppose user U creates a coin **c** and sends it to V. How do we ensure that U can no longer spend **c**.

Solution: Introduce ephemeral public-private address pairs.

- Coins attached to a public key can only be spent or transferred using the corresponding private key.

Version III: Transferring Spending Rights

Problem: Suppose user U creates a coin **c** and sends it to V. How do we ensure that U can no longer spend **c**.

Solution: Introduce ephemeral public-private address pairs.

- Coins attached to a public key can only be spent or transferred using the corresponding private key.
- A POUR transaction transfers the value of coins attached to U's public key to coins attached to V's public key.

Carnegie Mellon University

Version III: Transferring Spending Rights

Problem: Suppose user U creates a coin **c** and sends it to V. How do we ensure that U can no longer spend **c**.

Solution: Introduce ephemeral public-private address pairs.

- Coins attached to a public key can only be spent or transferred using the corresponding private key.
- A POUR transaction transfers the value of coins attached to U's public key to coins attached to V's public key.
- *Key Challenge:* The POUR transaction must hide the public keys.

Carnegie Mellon University

Creating Addresses

Suppose user U wants to create a new public-private address pair.

Creating Addresses

Suppose user U wants to create a new public-private address pair.

- Sample random secret key a_{sk}.

Carnegie Mellon University

Creating Addresses

Suppose user U wants to create a new public-private address pair.

- Sample random secret key a_{sk}.
- Using a_{sk} as a seed compute $a_{\mathsf{pk}} \leftarrow \mathsf{PRF}^{\mathsf{addr}}_{a_{\mathsf{sk}}}(0)$.

Carnegie Mellon University

Creating Addresses

Suppose user U wants to create a new public-private address pair.

- Sample random secret key a_{sk}.
- Using a_{sk} as a seed compute $a_{\mathsf{pk}} \leftarrow \mathsf{PRF}^{\mathsf{addr}}_{a_{\mathsf{sk}}}(0)$.
- Let the public-private address pair be

$$(a_{\mathsf{pk}}, a_{\mathsf{sk}})$$

Modifying Coin Generation

Suppose user U, with public-private address pair $(a_{\text{pk}}, a_{\text{sk}})$, wants to create a new coin.

- Sample sn.

Modifying Coin Generation

Suppose user U, with public-private address pair $(a_{\mathsf{pk}}, a_{\mathsf{sk}})$, wants to create a new coin.

- Sample sn.
- Sample s and compute commitment $\mathsf{cm} \leftarrow \mathsf{com}(v, a_{\mathsf{pk}}, \mathsf{sn}; s)$

Carnegie Mellon University

Modifying Coin Generation

Suppose user U, with public-private address pair $(a_{\mathsf{pk}}, a_{\mathsf{sk}})$, wants to create a new coin.

- Sample sn.
- Sample s and compute commitment $\mathsf{cm} \leftarrow \mathsf{com}(v, a_{\mathsf{pk}}, \mathsf{sn}; s)$
- Let the new private coin be

$$\mathbf{c} \leftarrow (a_{\mathsf{pk}}, v, \mathsf{sn}, s, \mathsf{cm}).$$

Carnegie Mellon University

Modifying Coin Generation

Suppose user U, with public-private address pair $(a_{\mathsf{pk}}, a_{\mathsf{sk}})$, wants to create a new coin.

- Sample sn.
- Sample s and compute commitment $\mathsf{cm} \leftarrow \mathsf{com}(v, a_{\mathsf{pk}}, \mathsf{sn}; s)$
- Let the new private coin be

$$\mathbf{c} \leftarrow (a_{\mathsf{pk}}, v, \mathsf{sn}, s, \mathsf{cm}).$$

Problem 1: In order to mint, cm needs to be opened to reveal v. However, this also reveals a_{pk} and sn.

Modifying Coin Generation

Suppose user U, with public-private address pair $(a_{\mathsf{pk}}, a_{\mathsf{sk}})$, wants to create a new coin.

- Sample sn.
- Sample s and compute commitment $\mathsf{cm} \leftarrow \mathsf{com}(v, a_{\mathsf{pk}}, \mathsf{sn}; s)$
- Let the new private coin be

$$\mathbf{c} \leftarrow (a_{\mathsf{pk}}, v, \mathsf{sn}, s, \mathsf{cm}).$$

Problem 1: In order to mint, cm needs to be opened to reveal v. However, this also reveals a_{pk} and sn.

Problem 2: If U knows sn it can track how the the coin is transferred on the network.

Modifying Coin Generation

Solution: Use nested commitments to bind values at different layers and PRFs to refresh sn in each transfer.

Modifying Coin Generation

Solution: Use nested commitments to bind values at different layers and PRFs to refresh sn in each transfer.

- Sample ρ and let sn $\leftarrow \mathsf{PRF}^{sn}_{\mathsf{a_{sk}}}(\rho)$.

Carnegie Mellon University

Modifying Coin Generation

Solution: Use nested commitments to bind values at different layers and PRFs to refresh sn in each transfer.

- Sample ρ and let $\mathsf{sn} \leftarrow \mathsf{PRF}^{sn}_{\mathsf{a_{sk}}}(\rho)$.
- Sample r and compute commitment $k \leftarrow \mathsf{com}(a_{\mathsf{pk}}, \rho; r)$.

Carnegie Mellon University

Modifying Coin Generation

Solution: Use nested commitments to bind values at different layers and PRFs to refresh sn in each transfer.

- Sample ρ and let $\mathsf{sn} \leftarrow \mathsf{PRF}^{sn}_{\mathsf{a_{sk}}}(\rho)$.
- Sample r and compute commitment $k \leftarrow \mathsf{com}(a_{\mathsf{pk}}, \rho; r)$.
- Sample s and compute commitment $\mathsf{cm} \leftarrow \mathsf{com}(v, k; s)$

Modifying Coin Generation

Solution: Use nested commitments to bind values at different layers and PRFs to refresh sn in each transfer.

- Sample ρ and let $\mathsf{sn} \leftarrow \mathsf{PRF}^{sn}_{\mathsf{a_{sk}}}(\rho)$.
- Sample r and compute commitment $k \leftarrow \mathsf{com}(a_{\mathsf{pk}}, \rho; r)$.
- Sample s and compute commitment $\mathsf{cm} \leftarrow \mathsf{com}(v, k; s)$
- Let the new private coin be

$$\mathbf{c} \leftarrow (a_{\mathsf{pk}}, v, \rho, r, s, \mathsf{cm}).$$

Carnegie Mellon University

Modifying Coin Generation

Solution: Use nested commitments to bind values at different layers and PRFs to refresh sn in each transfer.

- Sample ρ and let $\mathsf{sn} \leftarrow \mathsf{PRF}^{sn}_{\mathsf{a_{sk}}}(\rho)$.
- Sample r and compute commitment $k \leftarrow \mathsf{com}(a_{\mathsf{pk}}, \rho; r)$.
- Sample s and compute commitment $\mathsf{cm} \leftarrow \mathsf{com}(v, k; s)$
- Let the new private coin be

$$\mathbf{c} \leftarrow (a_{\mathsf{pk}}, v, \rho, r, s, \mathsf{cm}).$$

Now cm can be opened to reveal v but still hide sn and a_{pk}.

The Pour Operation

Suppose user U with keypair $(a_{\mathrm{pk}}^{\mathrm{old}}, a_{\mathrm{sk}}^{\mathrm{old}})$ wants to transfer $\mathbf{c}^{\mathrm{old}}$ to public keys $a_{\mathrm{pk},1}^{\mathrm{new}}$ and $a_{\mathrm{pk},2}^{\mathrm{new}}$.

Carnegie Mellon University

The Pour Operation

Suppose user U with keypair $(a_{\text{pk}}^{\text{old}}, a_{\text{sk}}^{\text{old}})$ wants to transfer $\mathbf{c}^{\text{old}}$ to public keys $a_{\text{pk},1}^{\text{new}}$ and $a_{\text{pk},2}^{\text{new}}$.

- U generates two coins $\mathbf{c}_1^{\text{new}}$ and $\mathbf{c}_2^{\text{new}}$ using $a_{\text{pk},1}^{\text{new}}$ and $a_{\text{pk},2}^{\text{new}}$ respectively.

Carnegie Mellon University

The Pour Operation

Suppose user U with keypair $(a_{\text{pk}}^{\text{old}}, a_{\text{sk}}^{\text{old}})$ wants to transfer $\mathbf{c}^{\text{old}}$ to public keys $a_{\text{pk},1}^{\text{new}}$ and $a_{\text{pk},2}^{\text{new}}$.

- U generates two coins $\mathbf{c}_1^{\text{new}}$ and $\mathbf{c}_2^{\text{new}}$ using $a_{\text{pk},1}^{\text{new}}$ and $a_{\text{pk},2}^{\text{new}}$ respectively.
- U writes a zkSNARK proof π asserting the POUR statement.

The Pour Operation

Suppose user U with keypair $(a_{\mathsf{pk}}^{\mathsf{old}}, a_{\mathsf{sk}}^{\mathsf{old}})$ wants to transfer $\mathbf{c}^{\mathsf{old}}$ to public keys $a_{\mathsf{pk},1}^{\mathsf{new}}$ and $a_{\mathsf{pk},2}^{\mathsf{new}}$.

- U generates two coins $\mathbf{c}_1^{\mathsf{new}}$ and $\mathbf{c}_2^{\mathsf{new}}$ using $a_{\mathsf{pk},1}^{\mathsf{new}}$ and $a_{\mathsf{pk},2}^{\mathsf{new}}$ respectively.
- U writes a zkSNARK proof π asserting the POUR statement.
- U broadcasts a pour transaction

$$\mathsf{tx}_{\mathsf{Pour}} = (\mathsf{rt}, \mathsf{sn}^{\mathsf{old}}, \mathsf{cm}_1^{\mathsf{new}}, \mathsf{cm}_2^{\mathsf{new}}, \pi).$$

The Pour Operation

Suppose user U with keypair $(a_{\mathrm{pk}}^{\mathrm{old}}, a_{\mathrm{sk}}^{\mathrm{old}})$ wants to transfer $\mathbf{c}^{\mathrm{old}}$ to public keys $a_{\mathrm{pk},1}^{\mathrm{new}}$ and $a_{\mathrm{pk},2}^{\mathrm{new}}$.

- U generates two coins $\mathbf{c}_1^{\mathrm{new}}$ and $\mathbf{c}_2^{\mathrm{new}}$ using $a_{\mathrm{pk},1}^{\mathrm{new}}$ and $a_{\mathrm{pk},2}^{\mathrm{new}}$ respectively.
- U writes a zkSNARK proof π asserting the POUR statement.
- U broadcasts a pour transaction

$$\mathrm{tx}_{\mathrm{Pour}} = (\mathrm{rt}, \mathrm{sn}^{\mathrm{old}}, \mathrm{cm}_1^{\mathrm{new}}, \mathrm{cm}_2^{\mathrm{new}}, \pi).$$

- The ledger accepts $\mathrm{tx}_{\mathrm{Pour}}$ if sn has not been seen before.

Carnegie Mellon University

The Pour Statement

Pour Statement

Carnegie Mellon University

The Pour Statement

Pour Statement

For **public** $(\mathsf{sn}, \mathsf{cm}_1^{\mathsf{new}}, \mathsf{cm}_2^{\mathsf{new}}, \mathsf{rt})$

The Pour Statement

Pour Statement

For **public** $(\mathsf{sn}, \mathsf{cm}_1^{\mathsf{new}}, \mathsf{cm}_2^{\mathsf{new}}, \mathsf{rt})$
I know **private** $(\mathbf{c}^{\mathsf{old}}, \mathbf{c}_1^{\mathsf{new}}, \mathbf{c}_2^{\mathsf{new}}, a_{\mathsf{sk}}^{\mathsf{old}}, \pi_{\mathsf{mk}})$

Carnegie Mellon University

The Pour Statement

Pour Statement

For **public** $(\mathsf{sn}, \mathsf{cm}_1^{\mathsf{new}}, \mathsf{cm}_2^{\mathsf{new}}, \mathsf{rt})$
I know **private** $(\mathbf{c}^{\mathsf{old}}, \mathbf{c}_1^{\mathsf{new}}, \mathbf{c}_2^{\mathsf{new}}, a_{\mathsf{sk}}^{\mathsf{old}}, \pi_{\mathsf{mk}})$
such that

The Pour Statement

Pour Statement

For **public** $(\mathsf{sn}, \mathsf{cm}_1^{\mathsf{new}}, \mathsf{cm}_2^{\mathsf{new}}, \mathsf{rt})$
I know **private** $(\mathbf{c}^{\mathsf{old}}, \mathbf{c}_1^{\mathsf{new}}, \mathbf{c}_2^{\mathsf{new}}, a_{\mathsf{sk}}^{\mathsf{old}}, \pi_{\mathsf{mk}})$
such that

- $\mathbf{c}.k = \mathsf{com}(\mathbf{c}.a_{\mathsf{pk}}, \mathbf{c}.\rho)$ and $\mathsf{cm} = \mathsf{com}(\mathbf{c}.v, \mathbf{c}.k)$.

Carnegie Mellon University

The Pour Statement

Pour Statement

For **public** $(\mathsf{sn}, \mathsf{cm}_1^{\mathsf{new}}, \mathsf{cm}_2^{\mathsf{new}}, \mathsf{rt})$
I know **private** $(\mathbf{c}^{\mathsf{old}}, \mathbf{c}_1^{\mathsf{new}}, \mathbf{c}_2^{\mathsf{new}}, a_{\mathsf{sk}}^{\mathsf{old}}, \pi_{\mathsf{mk}})$
such that

- $\mathbf{c}.k = \mathsf{com}(\mathbf{c}.a_{\mathsf{pk}}, \mathbf{c}.\rho)$ and $\mathsf{cm} = \mathsf{com}(\mathbf{c}.v, \mathbf{c}.k)$.
- The address of the old secret key matches the address found in the old coin.

Carnegie Mellon University

The Pour Statement

Pour Statement

For **public** $(\mathsf{sn}, \mathsf{cm}_1^{\mathsf{new}}, \mathsf{cm}_2^{\mathsf{new}}, \mathsf{rt})$
I know **private** $(\mathbf{c}^{\mathsf{old}}, \mathbf{c}_1^{\mathsf{new}}, \mathbf{c}_2^{\mathsf{new}}, a_{\mathsf{sk}}^{\mathsf{old}}, \pi_{\mathsf{mk}})$
such that

- $\mathbf{c}.k = \mathsf{com}(\mathbf{c}.a_{\mathsf{pk}}, \mathbf{c}.\rho)$ and $\mathsf{cm} = \mathsf{com}(\mathbf{c}.v, \mathbf{c}.k)$.
- The address of the old secret key matches the address found in the old coin.
- The serial number found in the old coin is computed correctly.

Carnegie Mellon University

The Pour Statement

Pour Statement

For **public** $(\mathsf{sn}, \mathsf{cm}_1^{\mathsf{new}}, \mathsf{cm}_2^{\mathsf{new}}, \mathsf{rt})$
I know **private** $(\mathbf{c}^{\mathsf{old}}, \mathbf{c}_1^{\mathsf{new}}, \mathbf{c}_2^{\mathsf{new}}, a_{\mathsf{sk}}^{\mathsf{old}}, \pi_{\mathsf{mk}})$
such that

- $\mathbf{c}.k = \mathsf{com}(\mathbf{c}.a_{\mathsf{pk}}, \mathbf{c}.\rho)$ and $\mathsf{cm} = \mathsf{com}(\mathbf{c}.v, \mathbf{c}.k)$.
- The address of the old secret key matches the address found in the old coin.
- The serial number found in the old coin is computed correctly.
- Merkle proof π_{mk} attests that $\mathbf{c}^{\mathsf{old}}.\mathsf{cm} \in \mathsf{Tree}(\mathsf{CMLIST})$.

Carnegie Mellon University

The Pour Statement

Pour Statement

For **public** $(\mathrm{sn}, \mathrm{cm}_1^{\mathrm{new}}, \mathrm{cm}_2^{\mathrm{new}}, \mathrm{rt})$
I know **private** $(\mathbf{c}^{\mathrm{old}}, \mathbf{c}_1^{\mathrm{new}}, \mathbf{c}_2^{\mathrm{new}}, a_{\mathrm{sk}}^{\mathrm{old}}, \pi_{\mathrm{mk}})$
such that

- $\mathbf{c}.k = \mathrm{com}(\mathbf{c}.a_{\mathrm{pk}}, \mathbf{c}.\rho)$ and $\mathrm{cm} = \mathrm{com}(\mathbf{c}.v, \mathbf{c}.k)$.
- The address of the old secret key matches the address found in the old coin.
- The serial number found in the old coin is computed correctly.
- Merkle proof π_{mk} attests that $\mathbf{c}^{\mathrm{old}}.\mathrm{cm} \in \mathrm{Tree}(\mathrm{CMLIST})$.
- $\mathbf{c}_1^{\mathrm{new}}.v + \mathbf{c}_2^{\mathrm{new}}.v = \mathbf{c}^{\mathrm{old}}.v$.

Carnegie Mellon University

The Pour Statement

Pour Statement

For **public** $(\mathsf{sn}, \mathsf{cm}_1^{\mathsf{new}}, \mathsf{cm}_2^{\mathsf{new}}, \mathsf{rt})$
I know **private** $(\mathbf{c}^{\mathsf{old}}, \mathbf{c}_1^{\mathsf{new}}, \mathbf{c}_2^{\mathsf{new}}, a_{\mathsf{sk}}^{\mathsf{old}}, \pi_{\mathsf{mk}})$
such that

- $\mathbf{c}.k = \mathsf{com}(\mathbf{c}.a_{\mathsf{pk}}, \mathbf{c}.\rho)$ and $\mathsf{cm} = \mathsf{com}(\mathbf{c}.v, \mathbf{c}.k)$.
- The address of the old secret key matches the address found in the old coin.
- The serial number found in the old coin is computed correctly.
- Merkle proof π_{mk} attests that $\mathbf{c}^{\mathsf{old}}.\mathsf{cm} \in \mathsf{Tree}(\mathsf{CMLIST})$.
- $\mathbf{c}_1^{\mathsf{new}}.v + \mathbf{c}_2^{\mathsf{new}}.v = \mathbf{c}^{\mathsf{old}}.v$.

Note that $\mathsf{tx}_{\mathsf{Pour}} = (\mathsf{rt}, \mathsf{sn}^{\mathsf{old}}, \mathsf{cm}_1^{\mathsf{new}}, \mathsf{cm}_2^{\mathsf{new}}, \pi)$ does not reveal values or public keys, and is therefore completely anonymous.

Carnegie Mellon University

How to Actually Send Coins

- Suppose user U posts $\mathsf{tx}_{\mathsf{Pour}} = (\mathsf{rt}, \mathsf{sn}^{\mathsf{old}}, \mathsf{cm}_1^{\mathsf{new}}, \mathsf{cm}_2^{\mathsf{new}}, \pi)$ on the ledger.

Carnegie Mellon University

How to Actually Send Coins

- Suppose user U posts $\mathsf{tx}_{\mathsf{Pour}} = (\mathsf{rt}, \mathsf{sn}^{\mathsf{old}}, \mathsf{cm}_1^{\mathsf{new}}, \mathsf{cm}_2^{\mathsf{new}}, \pi)$ on the ledger.
- User V can spend (or transfer) the value embedded in $\mathsf{cm}_i^{\mathsf{new}}$ so long as it can furnish the corresponding secret key and private coin $\mathbf{c}_i^{\mathsf{new}}$.

Carnegie Mellon University

How to Actually Send Coins

- Suppose user U posts $\mathsf{tx}_{\mathsf{Pour}} = (\mathsf{rt}, \mathsf{sn}^{\mathsf{old}}, \mathsf{cm}_1^{\mathsf{new}}, \mathsf{cm}_2^{\mathsf{new}}, \pi)$ on the ledger.
- User V can spend (or transfer) the value embedded in $\mathsf{cm}_i^{\mathsf{new}}$ so long as it can furnish the corresponding secret key and private coin $\mathbf{c}_i^{\mathsf{new}}$.

Problem: How does V actually get $\mathbf{c}_i^{\mathsf{new}}$?

How to Actually Send Coins

- Suppose user U posts $\mathrm{tx}_{\mathrm{Pour}} = (\mathrm{rt}, \mathrm{sn}^{\mathrm{old}}, \mathrm{cm}_1^{\mathrm{new}}, \mathrm{cm}_2^{\mathrm{new}}, \pi)$ on the ledger.
- User V can spend (or transfer) the value embedded in $\mathrm{cm}_i^{\mathrm{new}}$ so long as it can furnish the corresponding secret key and private coin $\mathbf{c}_i^{\mathrm{new}}$.

Problem: How does V actually get $\mathbf{c}_i^{\mathrm{new}}$?

Solution: Append public-key encryption keypairs to address keypairs.

Carnegie Mellon University

How to Actually Send Coins

- Suppose user U posts $\mathsf{tx}_{\mathsf{Pour}} = (\mathsf{rt}, \mathsf{sn}^{\mathsf{old}}, \mathsf{cm}_1^{\mathsf{new}}, \mathsf{cm}_2^{\mathsf{new}}, \pi)$ on the ledger.
- User V can spend (or transfer) the value embedded in $\mathsf{cm}_i^{\mathsf{new}}$ so long as it can furnish the corresponding secret key and private coin $\mathbf{c}_i^{\mathsf{new}}$.

Problem: How does V actually get $\mathbf{c}_i^{\mathsf{new}}$?

Solution: Append public-key encryption keypairs to address keypairs.

- U encrypts $\mathbf{c}_i^{\mathsf{new}}$ against V's public encryption key.

How to Actually Send Coins

- Suppose user U posts $\text{tx}_{\text{Pour}} = (\text{rt}, \text{sn}^{\text{old}}, \text{cm}_1^{\text{new}}, \text{cm}_2^{\text{new}}, \pi)$ on the ledger.
- User V can spend (or transfer) the value embedded in cm_i^{new} so long as it can furnish the corresponding secret key and private coin $\mathbf{c}_i^{\text{new}}$.

Problem: How does V actually get $\mathbf{c}_i^{\text{new}}$?

Solution: Append public-key encryption keypairs to address keypairs.

- U encrypts $\mathbf{c}_i^{\text{new}}$ against V's public encryption key.
- U appends the result to tx_{Pour}.

Carnegie Mellon University

Handling Public Outputs

- Construction so far allows for private minting, merging, and splitting of coins.

Handling Public Outputs

- Construction so far allows for private minting, merging, and splitting of coins.

Problem: How to lower ZEC back into the BTC?

Handling Public Outputs

- Construction so far allows for private minting, merging, and splitting of coins.

Problem: How to lower ZEC back into the BTC?

Solution: Modify the POUR statement.

Handling Public Outputs

- Construction so far allows for private minting, merging, and splitting of coins.

Problem: How to lower ZEC back into the BTC?

Solution: Modify the POUR statement.

- Allow user V to specify v_{pub} such that

$$v_1^{\text{new}} + v_2^{\text{new}} + v_{\text{pub}} = v^{\text{old}}$$

Handling Public Outputs

- Construction so far allows for private minting, merging, and splitting of coins.

Problem: How to lower ZEC back into the BTC?

Solution: Modify the POUR statement.

- Allow user V to specify v_{pub} such that

$$v_1^{\text{new}} + v_2^{\text{new}} + v_{\text{pub}} = v^{\text{old}}$$

- Additionally allow V to specify variable info that specifies a non-private address to deposit v_{pub} BTC.

Non-Malleability

Problem: How do we prevent miners from modifying info variable before posting transaction?

Non-Malleability

Problem: How do we prevent miners from modifying info variable before posting transaction?

Solution: Modify the POUR statement to include one-time digital signatures.

Carnegie Mellon University

Zcash

Zcash is worth \$2.08B, and is the canonical example of the commercial viability of advanced cryptography.

Zero-Knowledge Proofs beyond Zcash

Carnegie Mellon University

A Research Boom in zkSNARK Technology

A Research Boom in zkSNARK Technology

- MPC protocols designed just to decentralize the Zcash trusted setup.

A Research Boom in zkSNARK Technology

- MPC protocols designed just to decentralize the Zcash trusted setup.
- Initiated an entire line of research in zkSNARKs without a fully trusted setup [WTS+18, Set20, MBKM19].

A Research Boom in zkSNARK Technology

- MPC protocols designed just to decentralize the Zcash trusted setup.
- Initiated an entire line of research in zkSNARKs without a fully trusted setup [WTS+18, Set20, MBKM19].
- Revived interest in recursive zkSNARKs [Val08, BBB+18, KST21, BGH]

Clarifying Questions (Paraphrased)

- A minting transaction is a tuple (v, k, s, cm), where cm is $\mathrm{com}(v, k)$. How does a miner determine that the value v BTC is correct?
- POUR can split coins. Can it also merge them?
- Could the system be extended similar to the way Ethereum was created to allow for arbitrary private computation? [KMS$^+$16]

Carnegie Mellon University

Discussion Questions (Paraphrased)

- Is Zerocash ethical?
- What incentivizes the escrow pool to not abort the protocol and keep all the money?
- What is a good way of finding other people's address public keys in a privacy preserving manner?
- The authors mention that Zerocash could be deployed on top of any ledger, including a central bank's. How would such a deployment differ from a deployment over Bitcoin?
- By transferring a Bitcoin into a minted coin, the user needs to transfer it's bitcoin to a backing escrow pool first. Will this bring some security risks?

Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell.
Bulletproofs: Short proofs for confidential transactions and more.
In *IEEE S&P*, 2018.

Sean Bowe, Jack Grigg, and Daira Hopwood.
Halo: Recursive proof composition without a trusted setup.
IACR Cryptol. ePrint Arch., 2019.

Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou.
Hawk: The blockchain model of cryptography and privacy-preserving smart contracts.
In *IEEE S&P*, 2016.

Abhiram Kothapalli, Srinath Setty, and Ioanna Tzialla.
Nova: Recursive zero-knowledge arguments from folding schemes.
Cryptology ePrint Archive, Report 2021/370, 2021.
https://ia.cr/2021/370.

Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn.

Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings.
In *CCS*, 2019.

Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova.
Pinocchio: Nearly practical verifiable computation.
In *IEEE S&P*, 2013.

Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza.
Zerocash: Decentralized anonymous payments from Bitcoin.
In *IEEE S&P*, 2014.

Srinath Setty.
Spartan: Efficient and general-purpose zkSNARKs without trusted setup.
In *CRYPTO*, 2020.

Paul Valiant.
Incrementally verifiable computation or proofs of knowledge imply time/space efficiency.

In *TCC*, 2008.

Riad S Wahby, Ioanna Tzialla, Abhi Shelat, Justin Thaler, and Michael Walfish.
Doubly-efficient zkSNARKs without trusted setup.
In *IEEE S&P*, 2018.