ZEXE: Enabling Decentralized Private Computation

Sean Bowe	Zcash
Alessandro Chiesa	UC Berkeley
Matthew Green	JHU
Ian Miers	Cornell Tech
Pratyush Mishra	UC Berkeley
Howard Wu	UC Berkeley

PENCIL Workshop, 2019

ia.cr/2018/962   libzexe.org

Computing on distributed ledgers

Computing on distributed ledgers

- Today we have a number of systems for running general computations on distributed ledgers: Ethereum, Tezos, EOS…

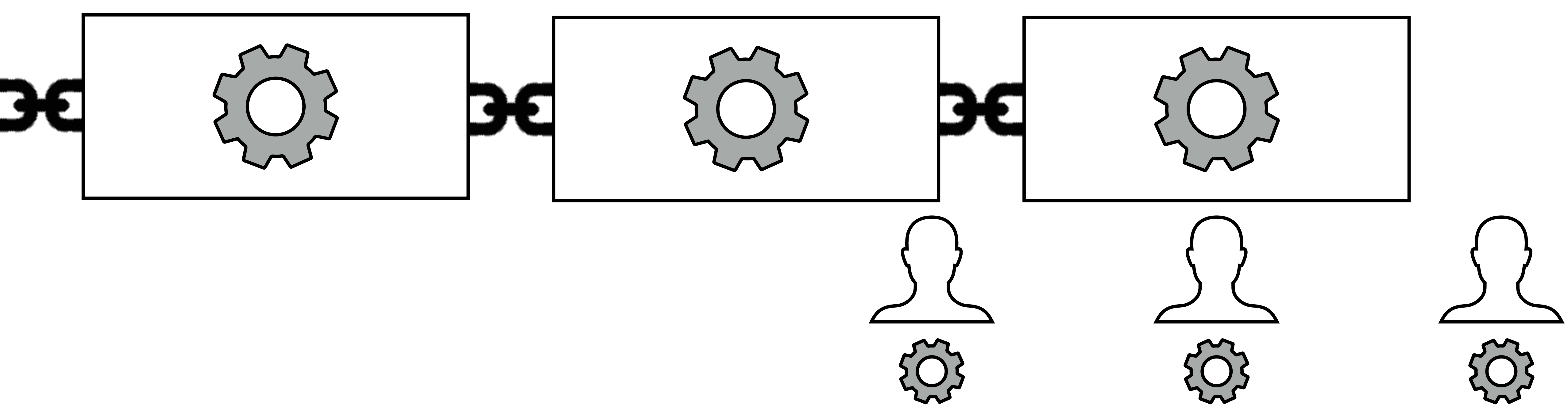
Computing on distributed ledgers

- Today we have a number of systems for running general computations on distributed ledgers: Ethereum, Tezos, EOS…
- These usually work by re-execution: every node re-executes every transaction.

Computing on distributed ledgers

- Today we have a number of systems for running general computations on distributed ledgers: Ethereum, Tezos, EOS…

- These usually work by re-execution: every node re-executes every transaction.



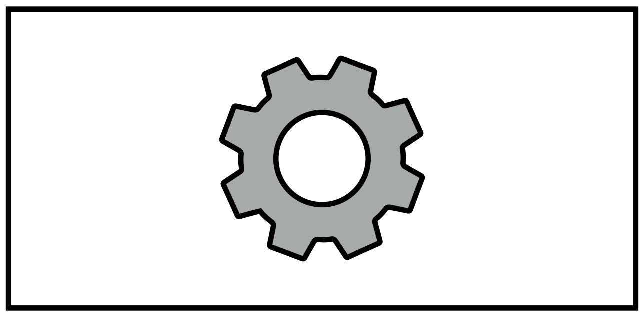
Scalability and privacy issues

Scalability

Privacy

Scalability and privacy issues

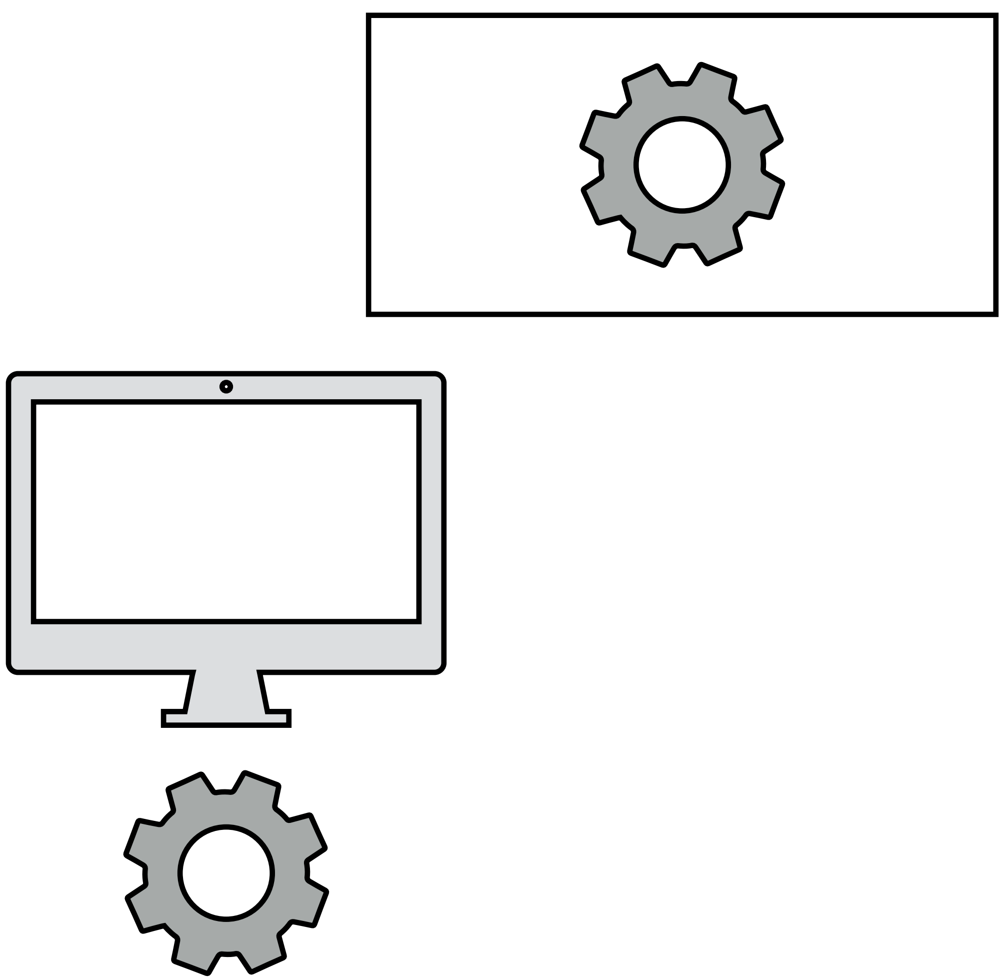
Scalability



Privacy

Scalability and privacy issues

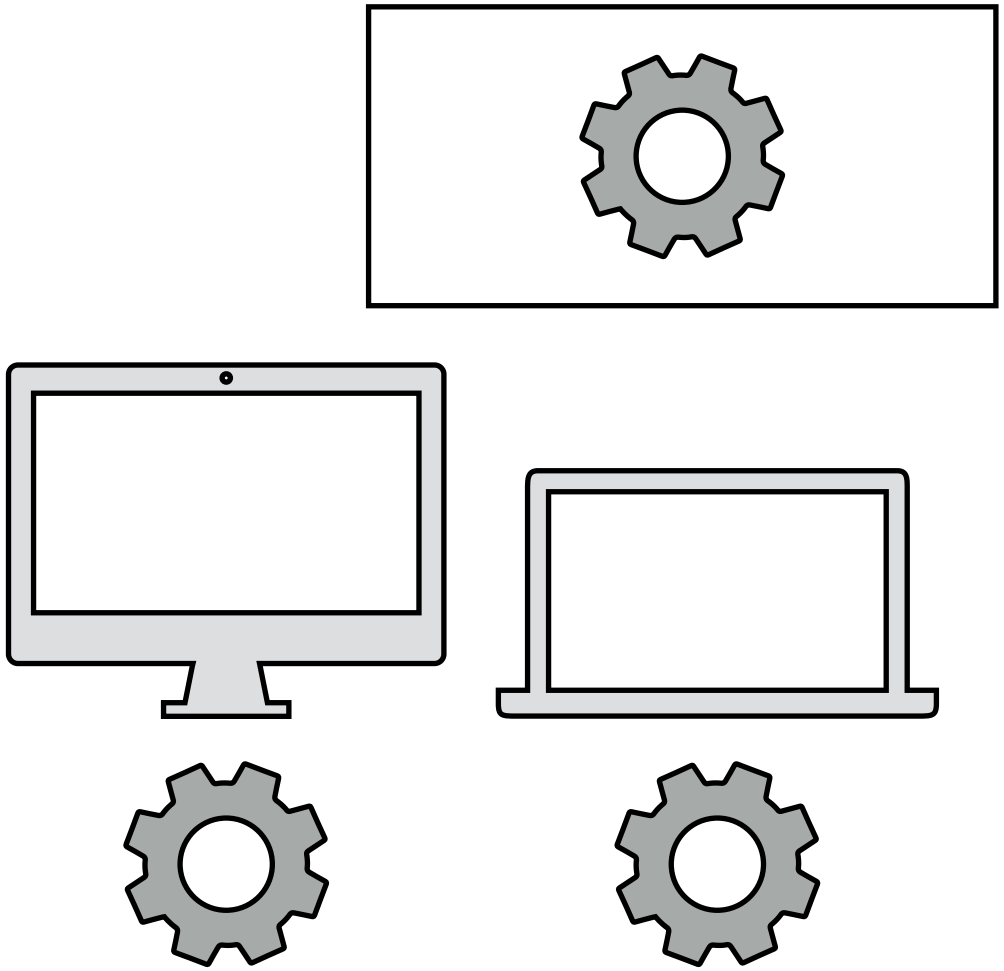
Scalability



Privacy

Scalability and privacy issues

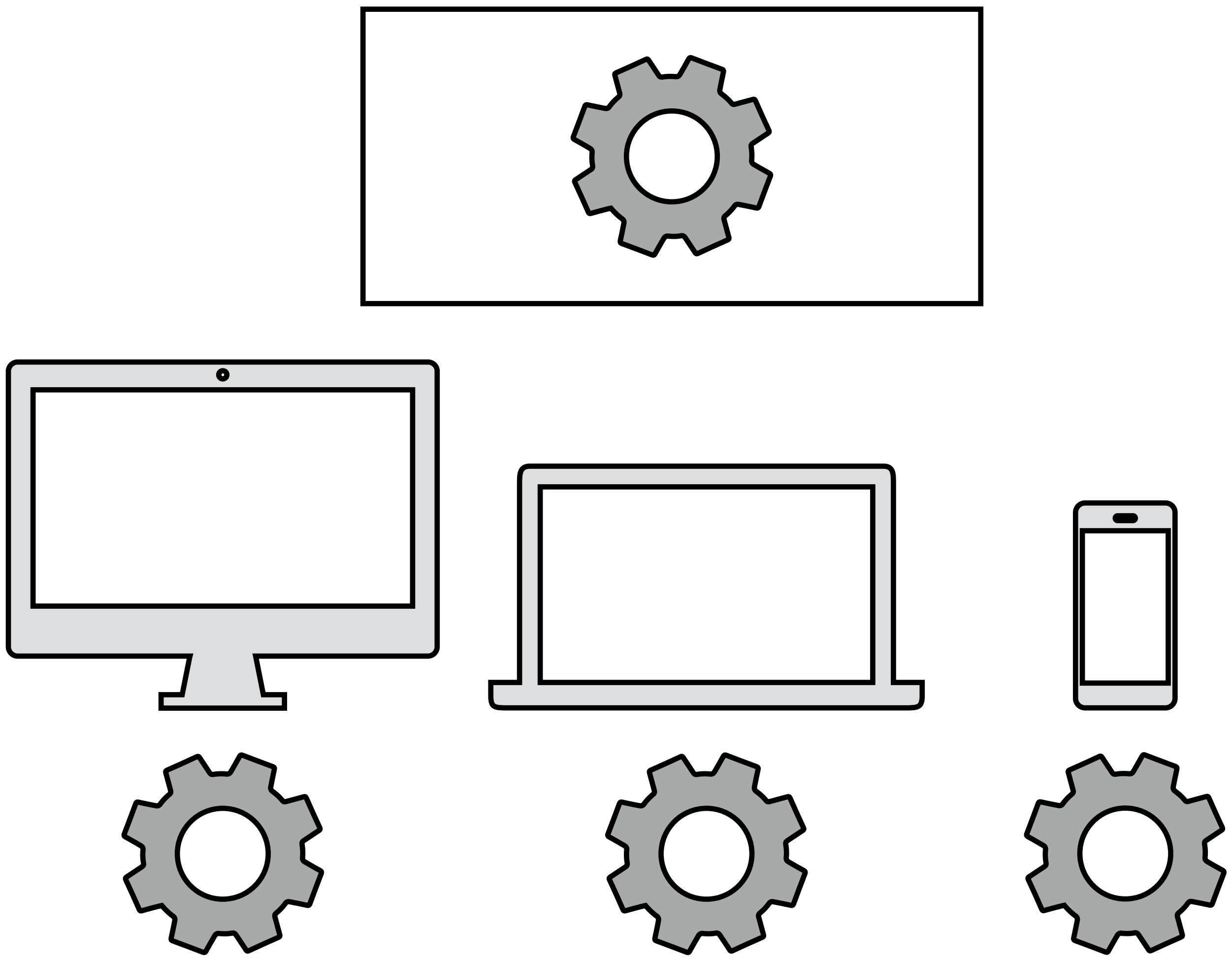
Scalability



Privacy

Scalability and privacy issues

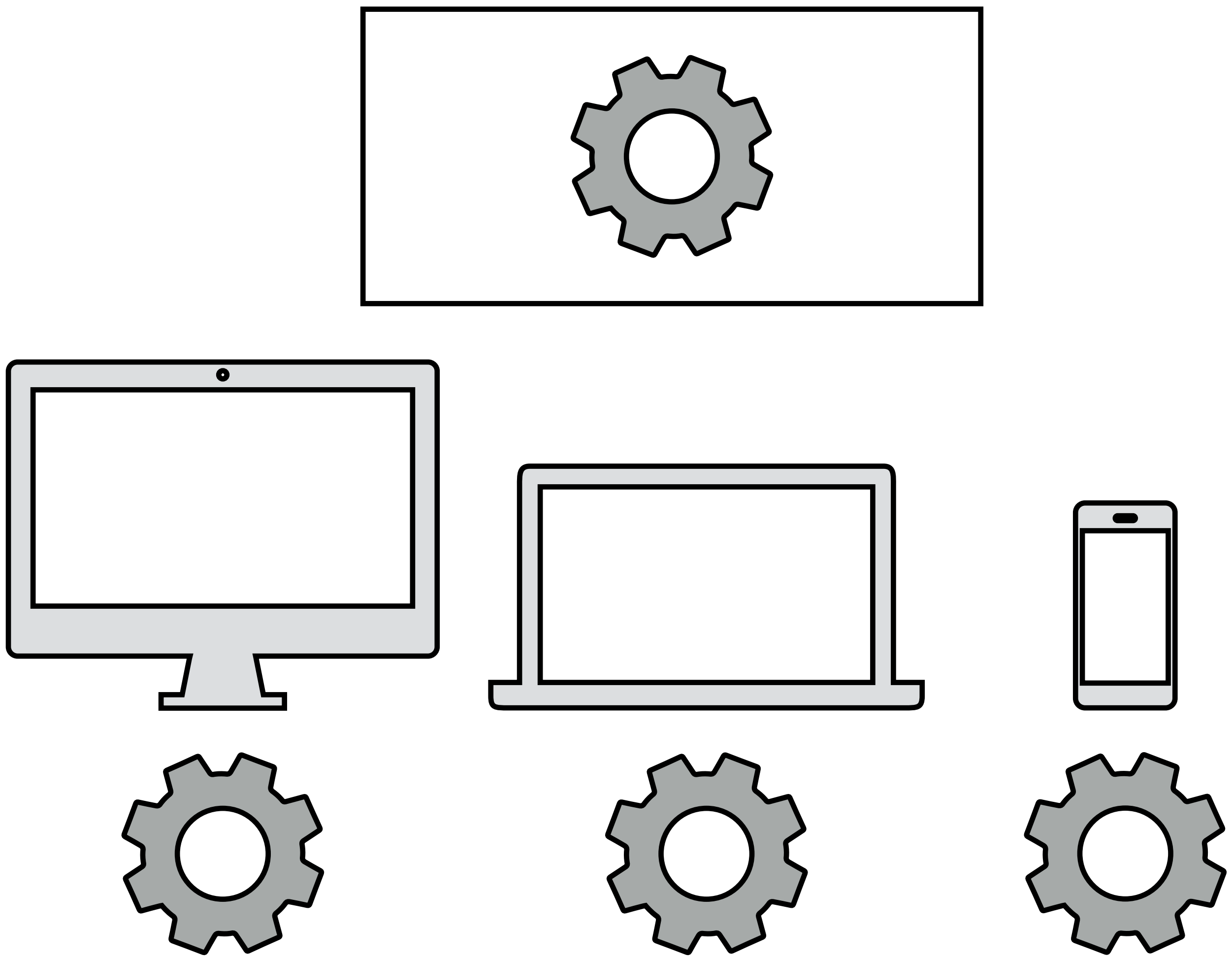
Scalability



Privacy

Scalability and privacy issues

Scalability

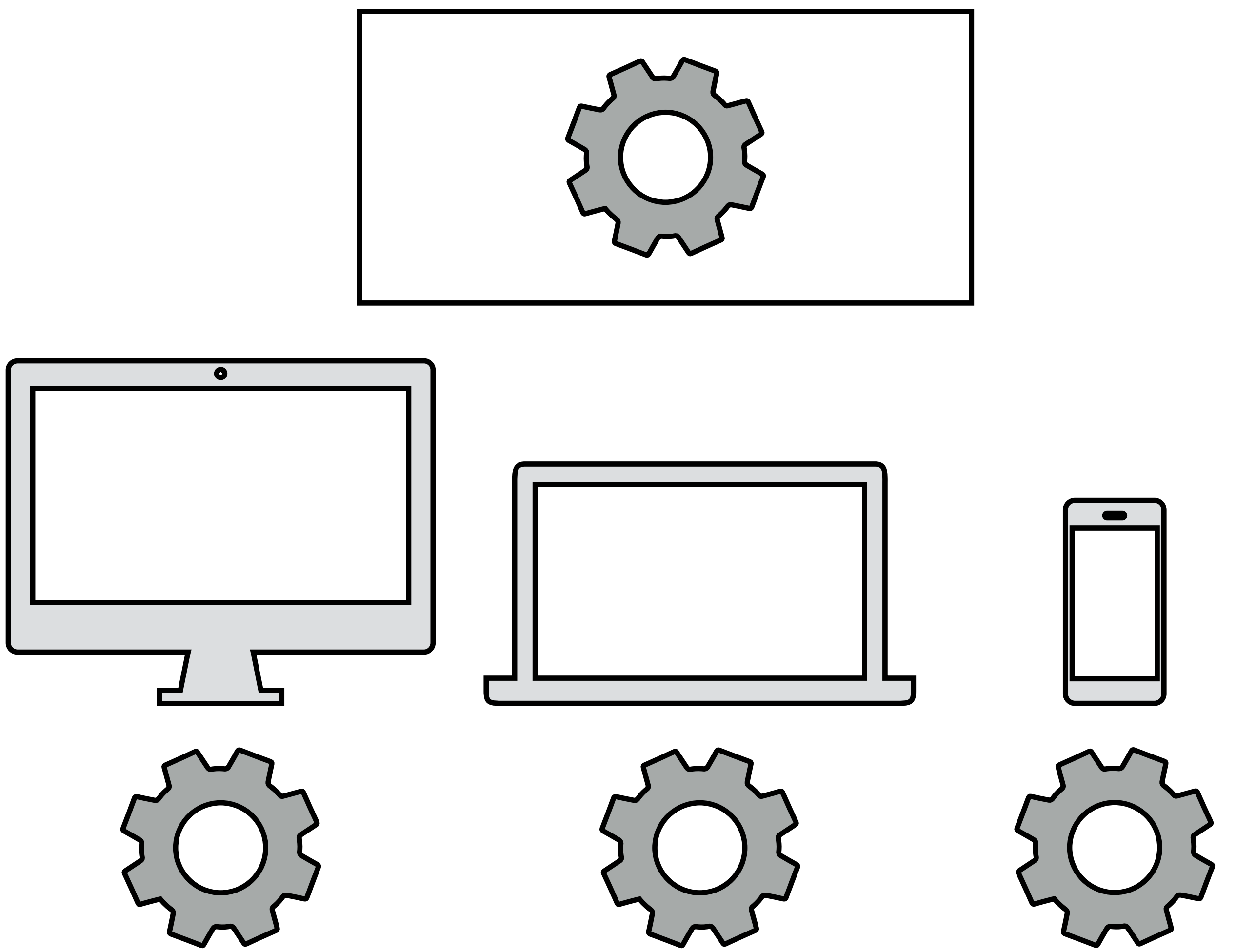


- Bottlenecked by weakest node

Privacy

Scalability and privacy issues

Scalability

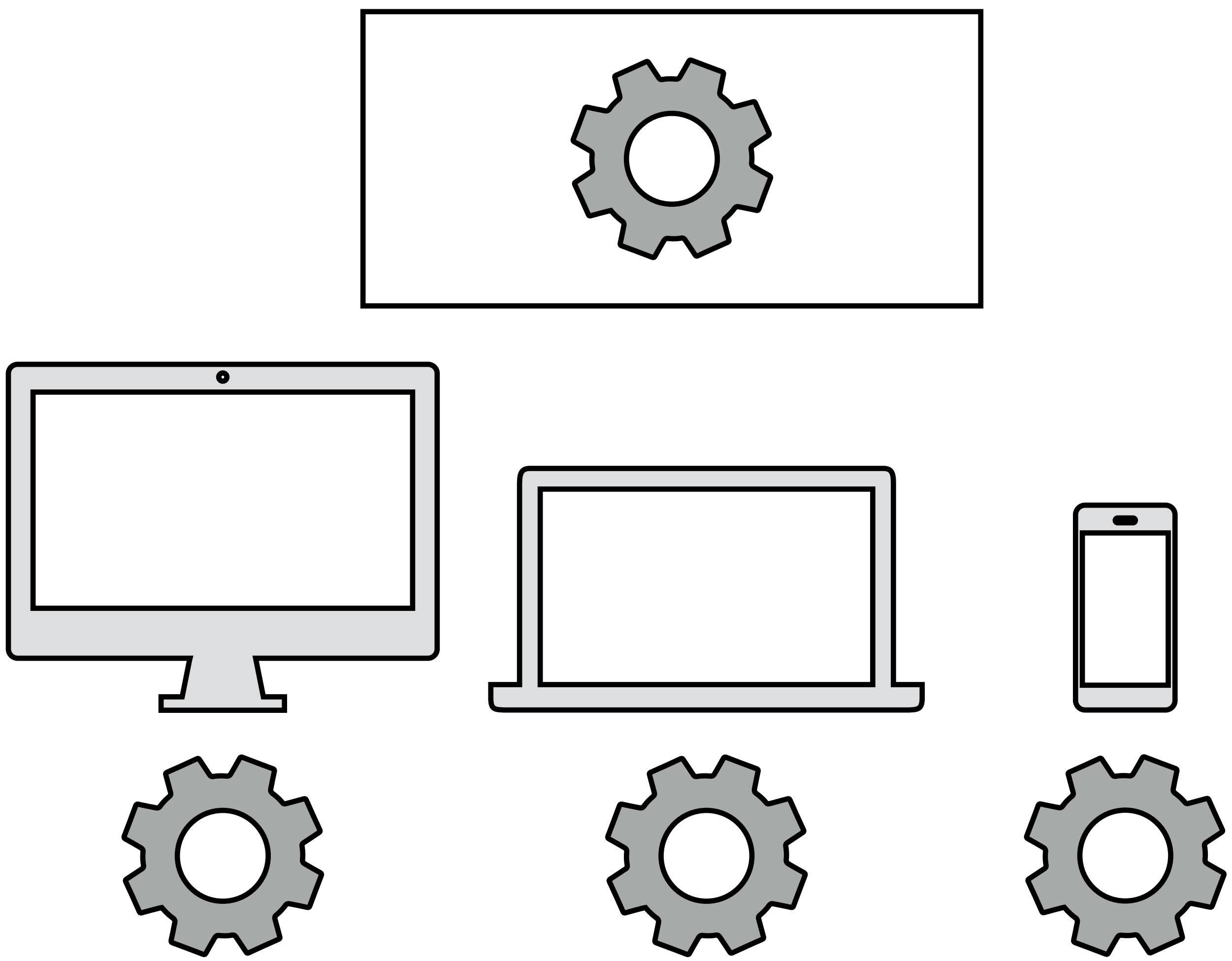


- Bottlenecked by weakest node
- Independent of consensus-layer improvements

Privacy

Scalability and privacy issues

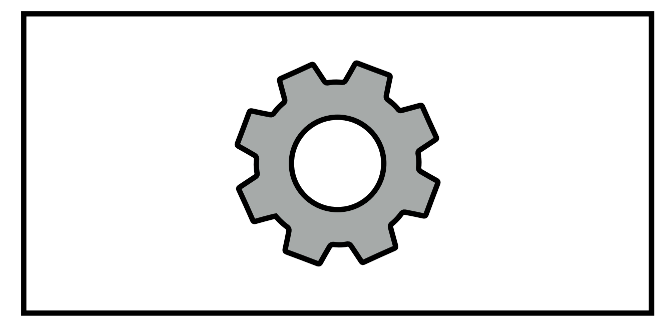
Scalability



- Bottlenecked by weakest node
- Independent of consensus-layer improvements

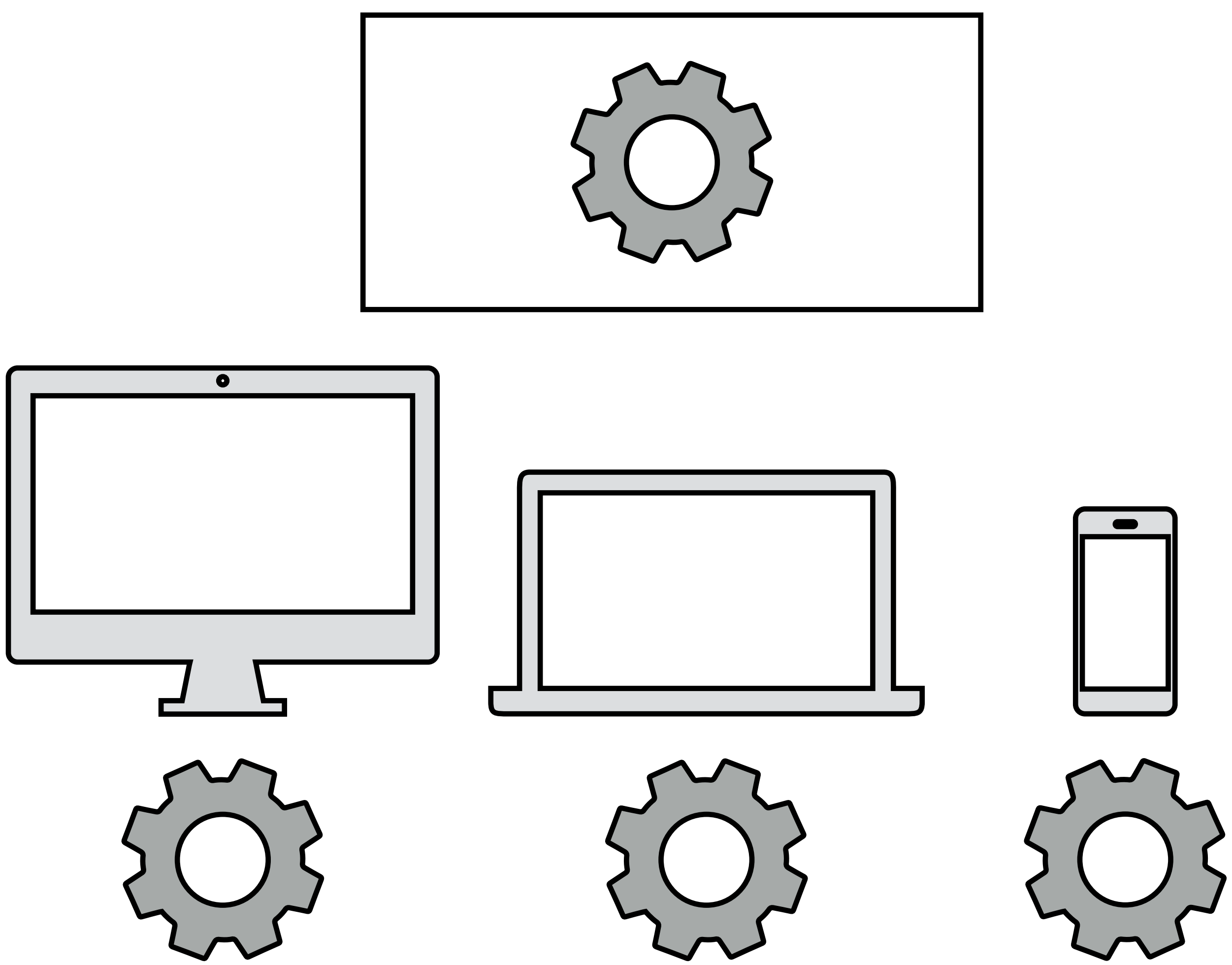
Privacy

Anyone can see:



Scalability and privacy issues

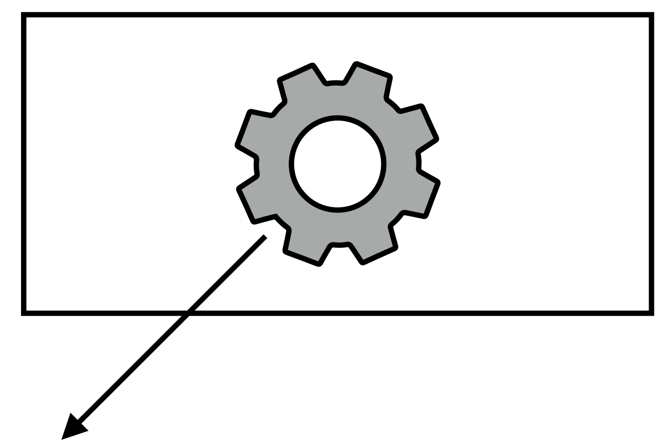
Scalability



- Bottlenecked by weakest node
- Independent of consensus-layer improvements

Privacy

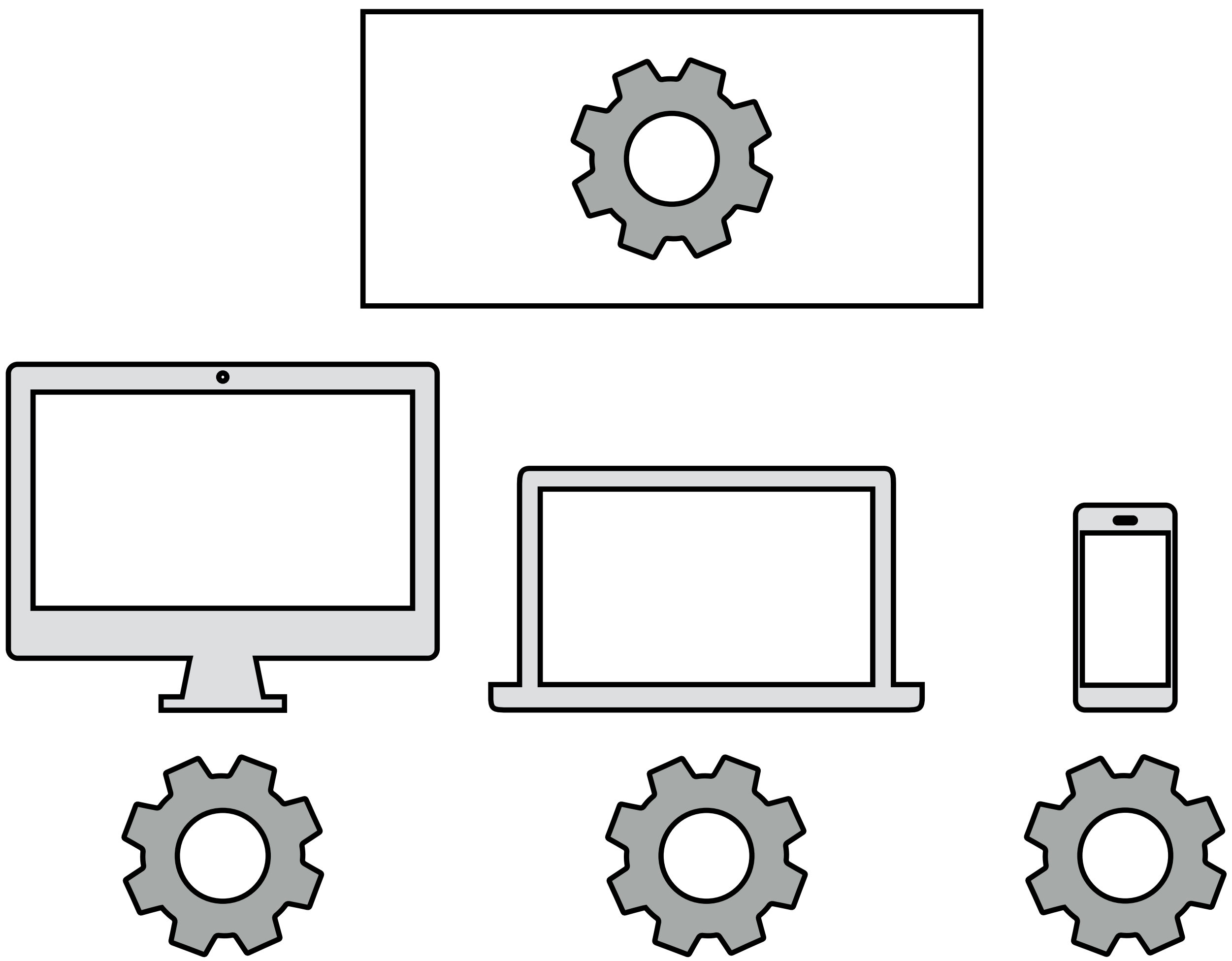
Anyone can see:



executed
program

Scalability and privacy issues

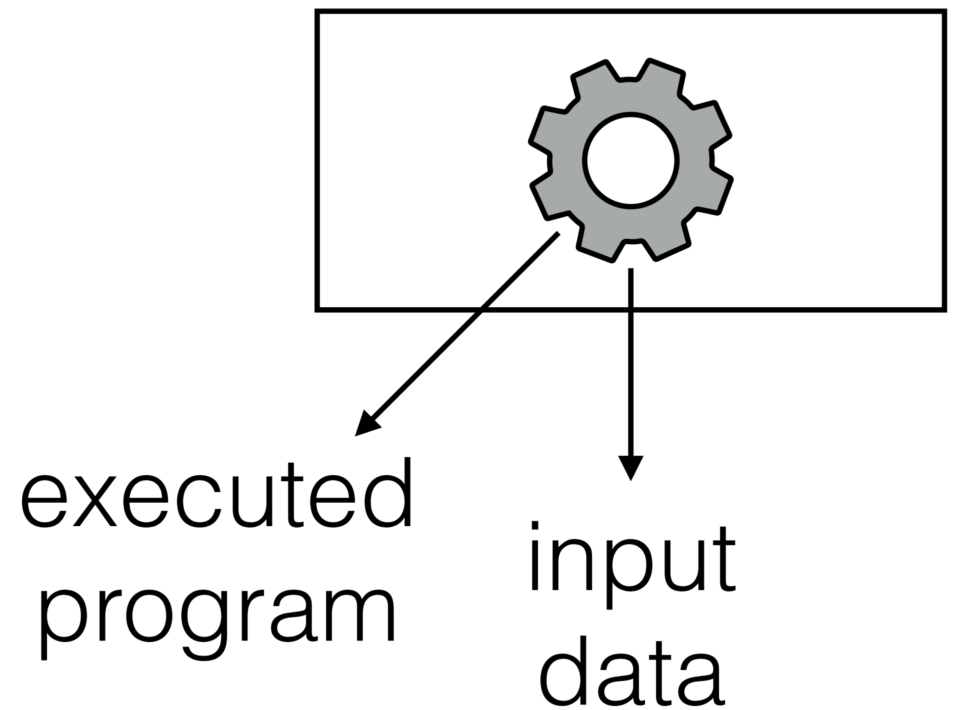
Scalability



- Bottlenecked by weakest node
- Independent of consensus-layer improvements

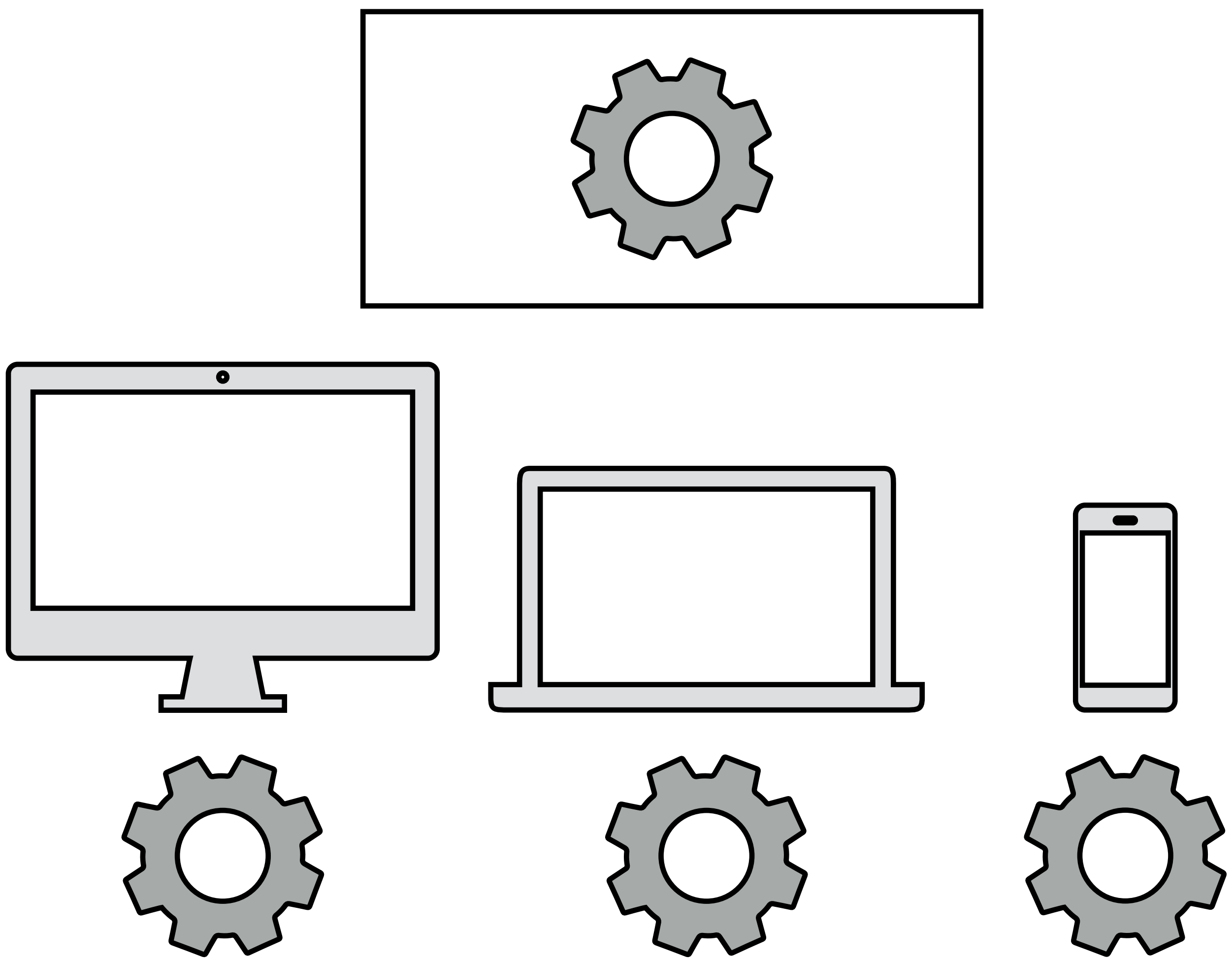
Privacy

Anyone can see:



Scalability and privacy issues

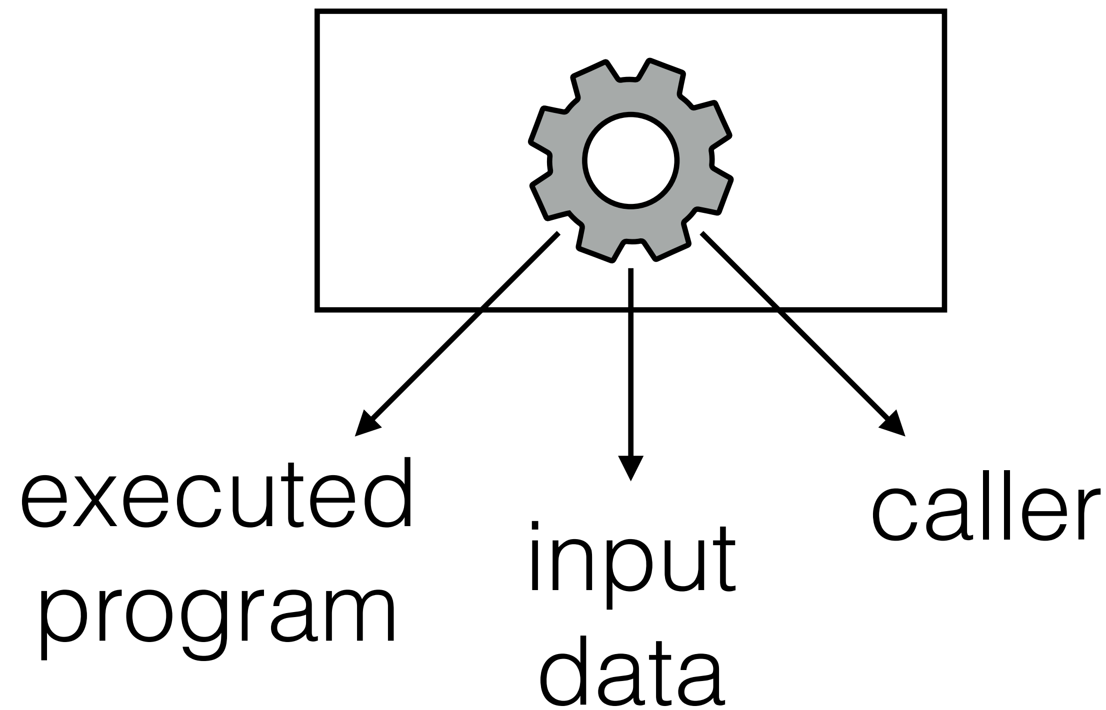
Scalability



- Bottlenecked by weakest node
- Independent of consensus-layer improvements

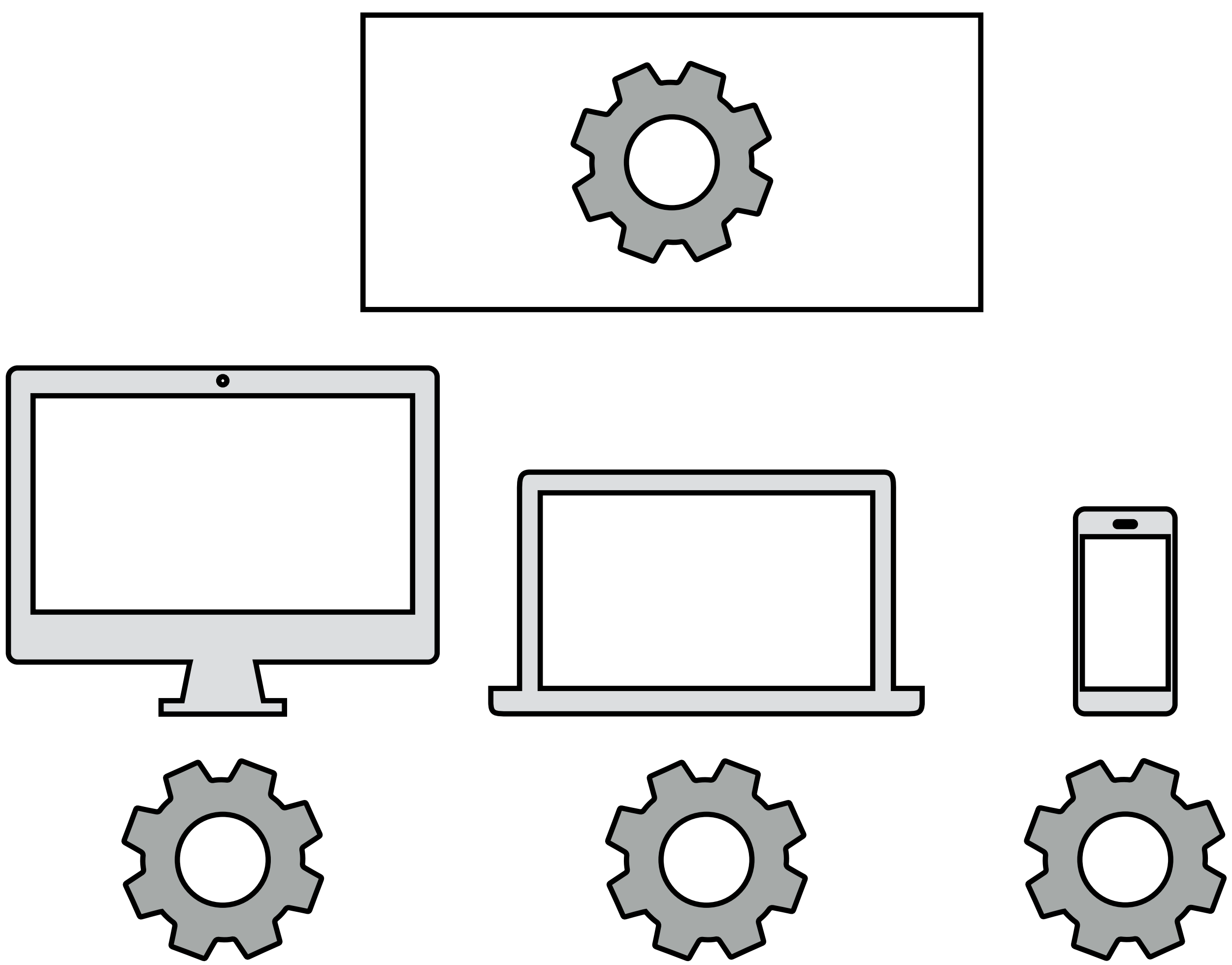
Privacy

Anyone can see:



Scalability and privacy issues

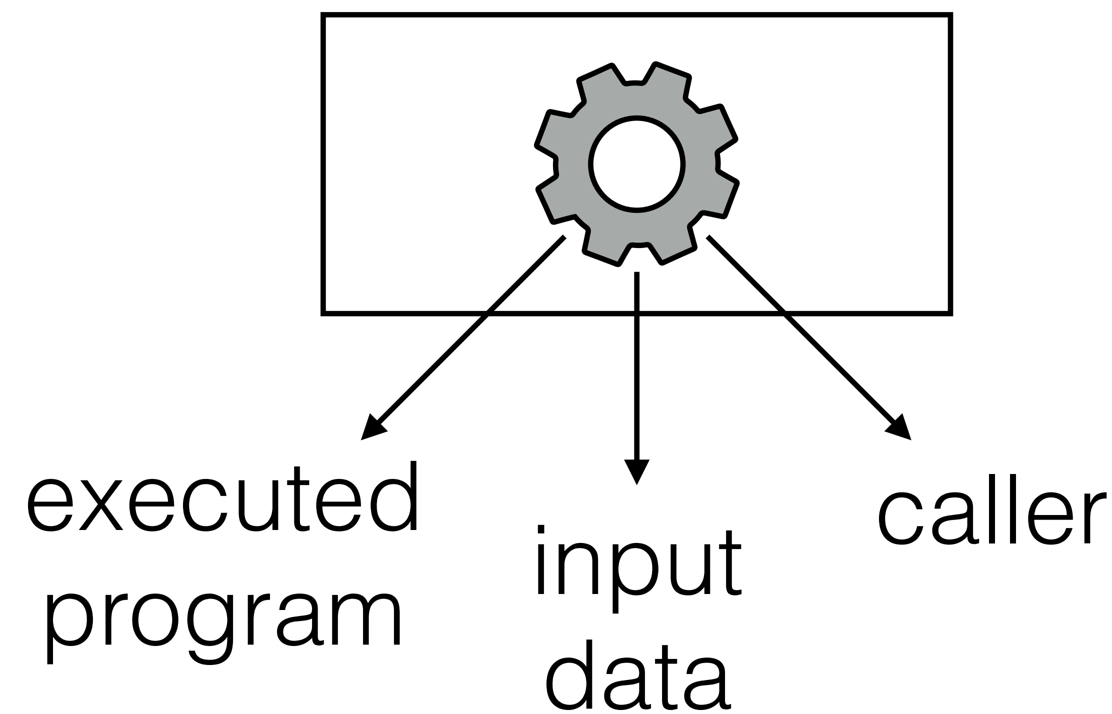
Scalability



- Bottlenecked by weakest node
- Independent of consensus-layer improvements

Privacy

Anyone can see:



- Permanently stored on the ledger

This work: **ZEXE**

This work: **ZEXE**

A ledger-based system that enables users to conduct offline computations and then publish transactions about these.

This work: **ZEXE**

A ledger-based system that enables users to conduct offline computations and then publish transactions about these.

Privacy: tx reveals no info about offline computation

This work: **ZEXE**

A ledger-based system that enables users to conduct offline computations and then publish transactions about these.

Privacy: tx reveals no info about offline computation

Succinctness: tx can be validated in poly(k) time

|tx|<1KB   Verify(tx) in <50ms

This work: **ZEXE**

A ledger-based system that enables users to conduct offline computations and then publish transactions about these.

Privacy: tx reveals no info about offline computation

Succinctness: tx can be validated in poly(k) time

|tx|<1KB   Verify(tx) in <50ms

Functionality:

This work: **ZEXE**

A ledger-based system that enables users to conduct offline computations and then publish transactions about these.

Privacy: tx reveals no info about offline computation

Succinctness: tx can be validated in poly(k) time

|tx|<1KB Verify(tx) in <50ms

Functionality:

- extensibility (user-defined functions)

This work: **ZEXE**

A ledger-based system that enables users to conduct offline computations and then publish transactions about these.

Privacy: tx reveals no info about offline computation

Succinctness: tx can be validated in poly(k) time

|tx|<1KB   Verify(tx) in <50ms

Functionality:

- extensibility (user-defined functions)
- isolation (malicious functions do not affect honest ones)

This work: **ZEXE**

A ledger-based system that enables users to conduct offline computations and then publish transactions about these.

Privacy: tx reveals no info about offline computation

Succinctness: tx can be validated in poly(k) time

|tx|<1KB   Verify(tx) in <50ms

Functionality:

- extensibility (user-defined functions)
- isolation (malicious functions do not affect honest ones)
- inter-process communication (functions can interact)

This work: **ZEXE**

A ledger-based system that enables users to conduct offline computations and then publish transactions about these.

Privacy: tx reveals no info about offline computation

Succinctness: tx can be validated in poly(k) time

|tx|<1KB   Verify(tx) in <50ms

Functionality:

- extensibility (user-defined functions)
- isolation (malicious functions do not affect honest ones)
- inter-process communication (functions can interact)

Example applications:

This work: **ZEXE**

A ledger-based system that enables users to conduct offline computations and then publish transactions about these.

Privacy: tx reveals no info about offline computation

Succinctness: tx can be validated in poly(k) time

|tx|<1KB Verify(tx) in <50ms

Functionality:

- extensibility (user-defined functions)
- isolation (malicious functions do not affect honest ones)
- inter-process communication (functions can interact)

Example applications:

- private user-defined assets

This work: **ZEXE**

A ledger-based system that enables users to conduct offline computations and then publish transactions about these.

Privacy: tx reveals no info about offline computation

Succinctness: tx can be validated in poly(k) time

|tx|<1KB Verify(tx) in <50ms

Functionality:

- extensibility (user-defined functions)
- isolation (malicious functions do not affect honest ones)
- inter-process communication (functions can interact)

Example applications:

- private user-defined assets
- private stablecoins

This work: **ZEXE**

A ledger-based system that enables users to conduct offline computations and then publish transactions about these.

Privacy: tx reveals no info about offline computation

Succinctness: tx can be validated in poly(k) time

|tx|<1KB Verify(tx) in <50ms

Functionality:

- extensibility (user-defined functions)
- isolation (malicious functions do not affect honest ones)
- inter-process communication (functions can interact)

Example applications:

- private user-defined assets
- private stablecoins
- private decentralized exchanges

This talk: **ZEXE**

This talk: ~~**ZEXE**~~

Will use ideas from ZEXE to construct

This talk: ~~**ZEXE**~~

Will use ideas from ZEXE to construct

private user-defined assets

This talk: ~~**ZEXE**~~

Will use ideas from ZEXE to construct

private user-defined assets
private decentralized exchanges

This talk: ~~**ZEXE**~~

Will use ideas from ZEXE to construct

private user-defined assets
private decentralized exchanges
private stablecoins

Trading digital assets

Trading digital assets

- Custom, user-defined assets are most successful application of smart contract systems.

Trading digital assets

- Custom, user-defined assets are most successful application of smart contract systems.

- Many different kinds of assets such as ERC-20, ERC-721 tokens and stablecoins. Different properties and controlling authorities.

Trading digital assets

- Custom, user-defined assets are most successful application of smart contract systems.

- Many different kinds of assets such as ERC-20, ERC-721 tokens and stablecoins. Different properties and controlling authorities.

- How to trade such assets?

Trading digital assets

- Custom, user-defined assets are most successful application of smart contract systems.

- Many different kinds of assets such as ERC-20, ERC-721 tokens and stablecoins. Different properties and controlling authorities.

- How to trade such assets?

Centralized exchanges

Trading digital assets

- Custom, user-defined assets are most successful application of smart contract systems.

- Many different kinds of assets such as ERC-20, ERC-721 tokens and stablecoins. Different properties and controlling authorities.

- How to trade such assets?

Centralized exchanges

- **Users give up custody of assets.**

Trading digital assets

- Custom, user-defined assets are most successful application of smart contract systems.

- Many different kinds of assets such as ERC-20, ERC-721 tokens and stablecoins. Different properties and controlling authorities.

- How to trade such assets?

Centralized exchanges

- **Users give up custody of assets.**
  - **Can lead to loss of funds due to fraud/breach/error**

Trading digital assets

- Custom, user-defined assets are most successful application of smart contract systems.

- Many different kinds of assets such as ERC-20, ERC-721 tokens and stablecoins. Different properties and controlling authorities.

- How to trade such assets?

Centralized exchanges

- Users give up custody of assets.

  - Can lead to loss of funds due to fraud/breach/error

+ Efficient: infrequent on-chain activity

Trading digital assets

- Custom, user-defined assets are most successful application of smart contract systems.
- Many different kinds of assets such as ERC-20, ERC-721 tokens and stablecoins. Different properties and controlling authorities.
- How to trade such assets?

Centralized exchanges

- **Users give up custody of assets.**
  - **Can lead to loss of funds due to fraud/breach/error**

\+ **Efficient: infrequent on-chain activity**

\+ **(Somewhat) private: only exchange learns trade details**

Trading digital assets

- Custom, user-defined assets are most successful application of smart contract systems.
- Many different kinds of assets such as ERC-20, ERC-721 tokens and stablecoins. Different properties and controlling authorities.
- How to trade such assets?

Centralized exchanges

- **Users give up custody of assets.**
  - **Can lead to loss of funds due to fraud/breach/error**

\+ **Efficient: infrequent on-chain activity**

\+ **(Somewhat) private: only exchange learns trade details**

Decentralized **exchanges**

Trading digital assets

- Custom, user-defined assets are most successful application of smart contract systems.
- Many different kinds of assets such as ERC-20, ERC-721 tokens and stablecoins. Different properties and controlling authorities.
- How to trade such assets?

Centralized exchanges

- **Users give up custody of assets.**
  - **Can lead to loss of funds due to fraud/breach/error**

+ **Efficient: infrequent on-chain activity**

+ **(Somewhat) private: only exchange learns trade details**

***Decentralized* exchanges**

+ **Users retain custody of assets**

Trading digital assets

- Custom, user-defined assets are most successful application of smart contract systems.
- Many different kinds of assets such as ERC-20, ERC-721 tokens and stablecoins. Different properties and controlling authorities.
- How to trade such assets?

Centralized exchanges

- **Users give up custody of assets.**
  - **Can lead to loss of funds due to fraud/breach/error**

+ **Efficient: infrequent on-chain activity**

+ **(Somewhat) private: only exchange learns trade details**

***Decentralized* exchanges**

+ **Users retain custody of assets**

- **Inefficient: on-chain activity for all trades**

Trading digital assets

- Custom, user-defined assets are most successful application of smart contract systems.
- Many different kinds of assets such as ERC-20, ERC-721 tokens and stablecoins. Different properties and controlling authorities.
- How to trade such assets?

Centralized exchanges	***Decentralized* exchanges**
- Users give up custody of assets. - Can lead to loss of funds due to fraud/breach/error	+ Users retain custody of assets
+ Efficient: infrequent on-chain activity	- Inefficient: on-chain activity for all trades
+ (Somewhat) private: only exchange learns trade details	- Non-private: any observer can see trade details

Limitations of DEXs

Limitations of DEXs

Privacy:

Limitations of DEXs

Privacy:

- Every trade goes on the ledger.

Limitations of DEXs

Privacy:

- Every trade goes on the ledger.
- Transactions reveal participating parties, assets and values.

Limitations of DEXs

Privacy:

- Every trade goes on the ledger.
- Transactions reveal participating parties, assets and values.

Limitations of DEXs

Privacy:

- Every trade goes on the ledger.
- Transactions reveal participating parties, assets and values.

User financial privacy

Limitations of DEXs

Privacy:

- Every trade goes on the ledger.
- Transactions reveal participating parties, assets and values.

User financial privacy

Trading history of users is public on the ledger, which reduces fungibility and can reveal secrets like trading patterns and techniques.

Limitations of DEXs

Privacy:

- Every trade goes on the ledger.
- Transactions reveal participating parties, assets and values.

User financial privacy

Trading history of users is public on the ledger, which reduces fungibility and can reveal secrets like trading patterns and techniques.

Frontrunning

Miners' privileged network position enables them to see big orders before others, allowing them to place their own orders before prices change.

Limitations of DEXs

Privacy:

- Every trade goes on the ledger.
- Transactions reveal participating parties, assets and values.

User financial privacy

Trading history of users is public on the ledger, which reduces fungibility and can reveal secrets like trading patterns and techniques.

Frontrunning

Miners’ privileged network position enables them to see big orders before others, allowing them to place their own orders before prices change.

[BDJT17, BBDJLZ17, DGKLZBBJ19]

Private user-defined assets

Private user-defined assets

Private DEX

Private user-defined assets

Private DEX

Private stablecoins

Private user-defined assets

Private DEX

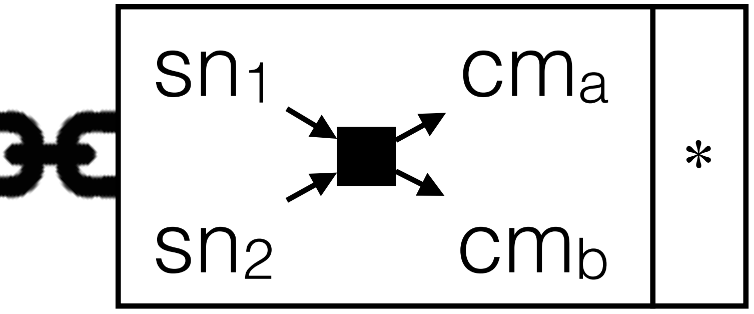
Private stablecoins

The Zerocash Paradigm [BCGGMTV14]

Each transaction consumes old coins and creates new coins.
The senders, receivers, and values are provably hidden.

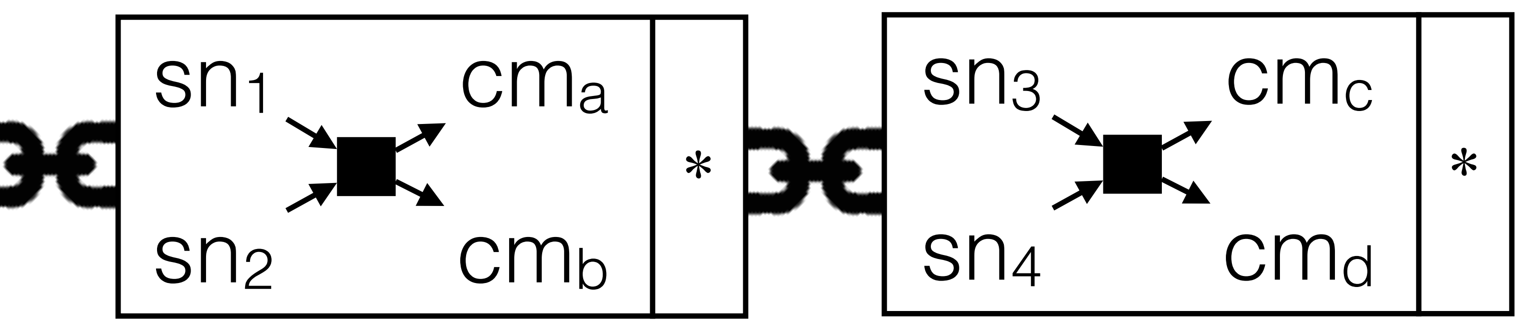
The Zerocash Paradigm [BCGGMTV14]

Each transaction consumes old coins and creates new coins.
The senders, receivers, and values are provably hidden.



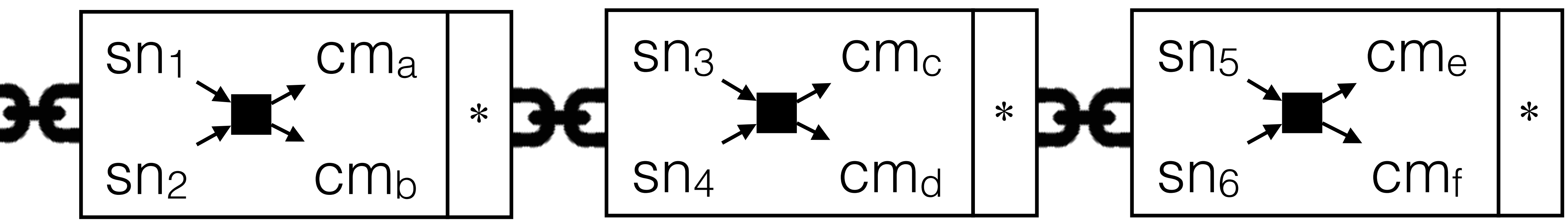
The Zerocash Paradigm [BCGGMTV14]

Each transaction consumes old coins and creates new coins.
The senders, receivers, and values are provably hidden.



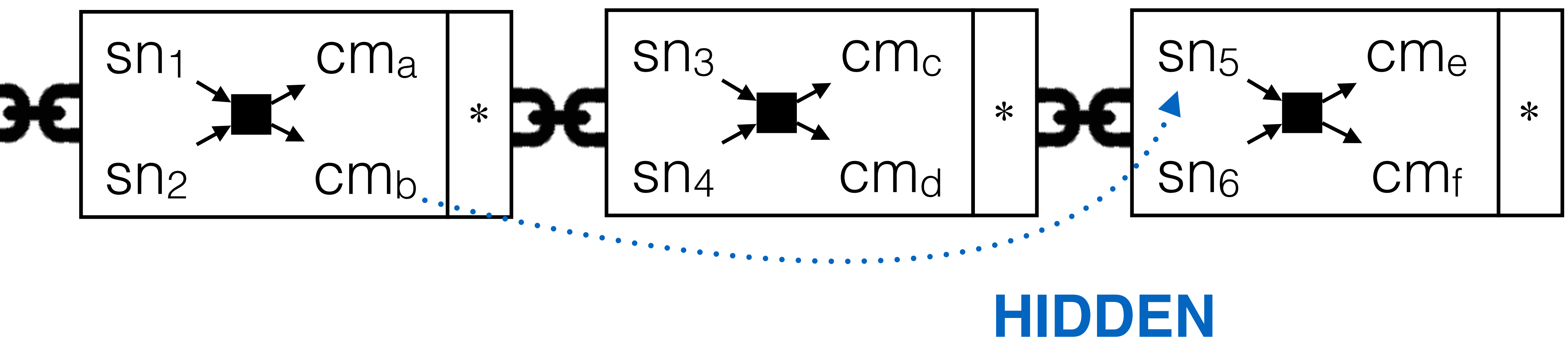
The Zerocash Paradigm [BCGGMTV14]

Each transaction consumes old coins and creates new coins.
The senders, receivers, and values are provably hidden.



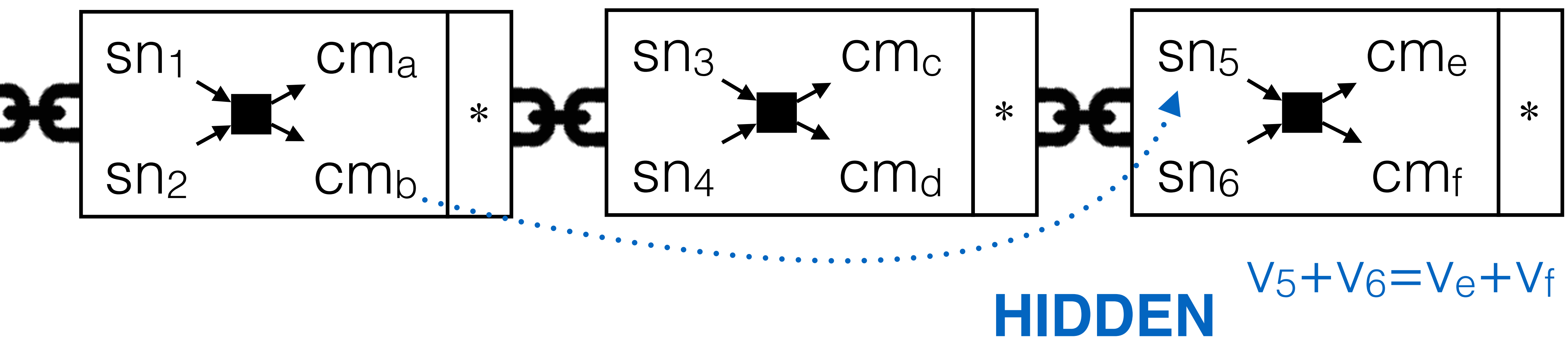
The Zerocash Paradigm [BCGGMTV14]

Each transaction consumes old coins and creates new coins.
The senders, receivers, and values are provably hidden.



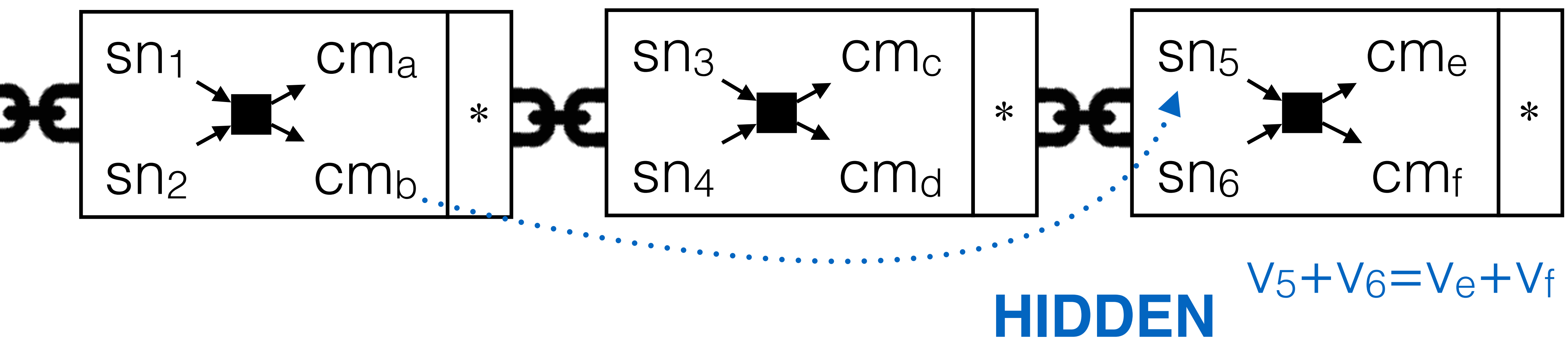
The Zerocash Paradigm [BCGGMTV14]

Each transaction consumes old coins and creates new coins.
The senders, receivers, and values are provably hidden.



The Zerocash Paradigm [BCGGMTV14]

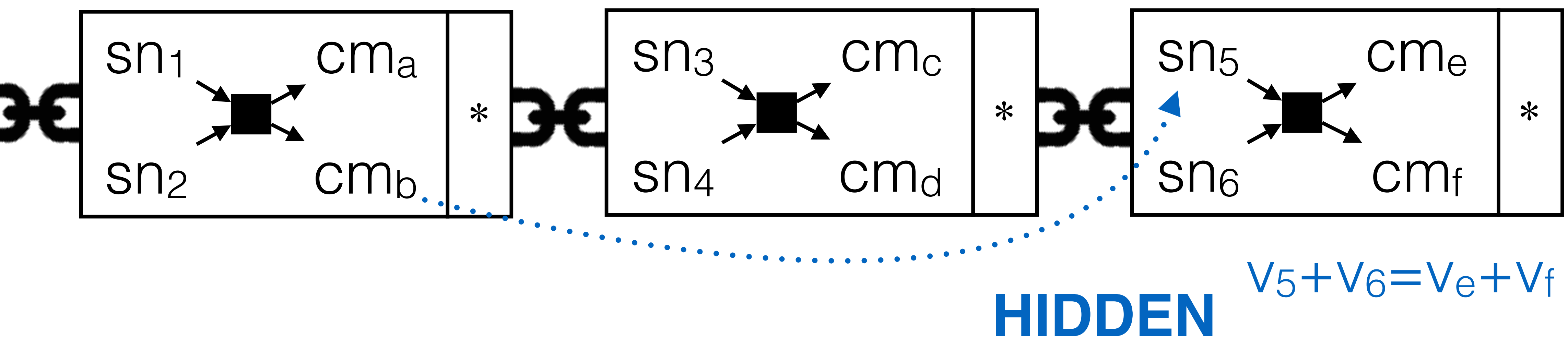
Each transaction consumes old coins and creates new coins.
The senders, receivers, and values are provably hidden.



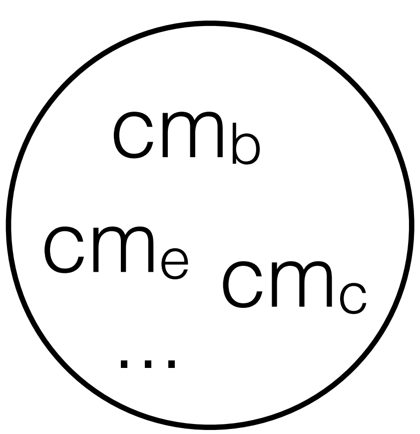
all created coins
(commitments)

The Zerocash Paradigm [BCGGMTV14]

Each transaction consumes old coins and creates new coins.
The senders, receivers, and values are provably hidden.

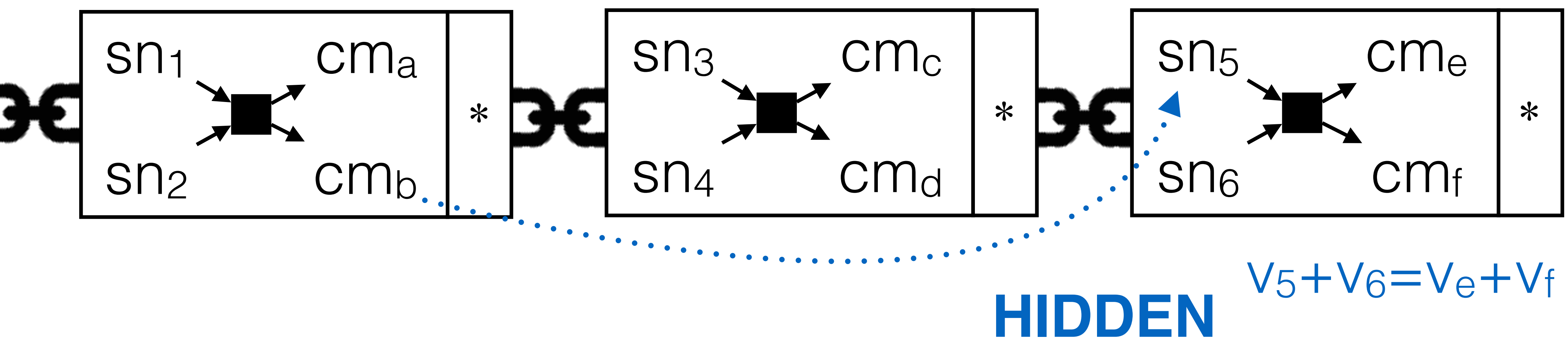


all created coins (commitments)

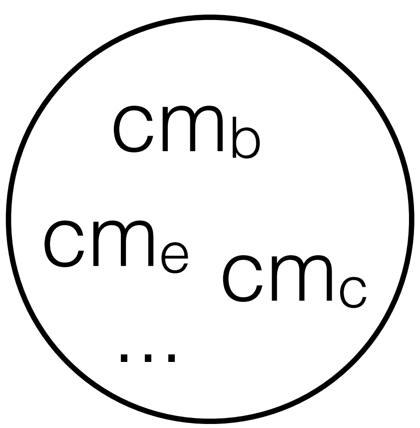


The Zerocash Paradigm [BCGGMTV14]

Each transaction consumes old coins and creates new coins.
The senders, receivers, and values are provably hidden.



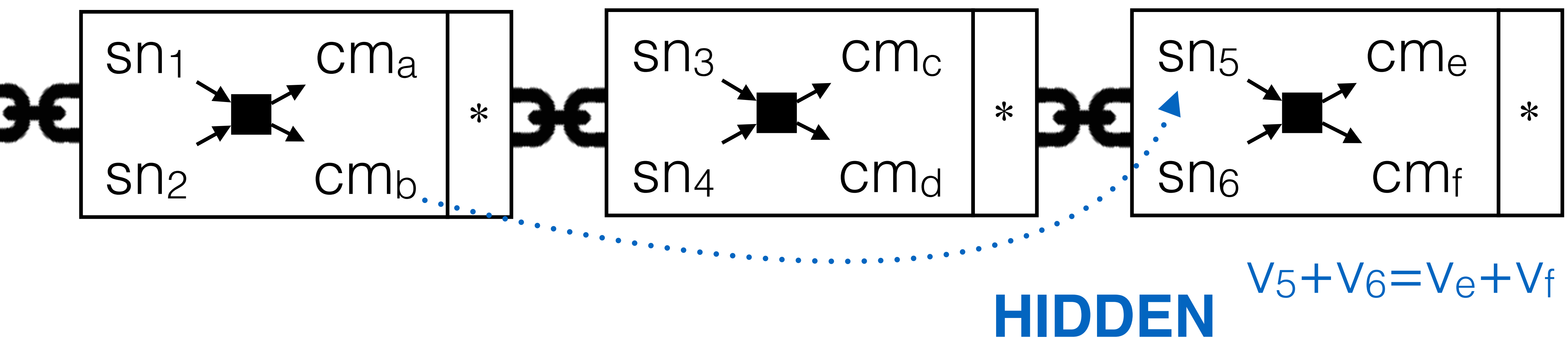
all created coins
(commitments)



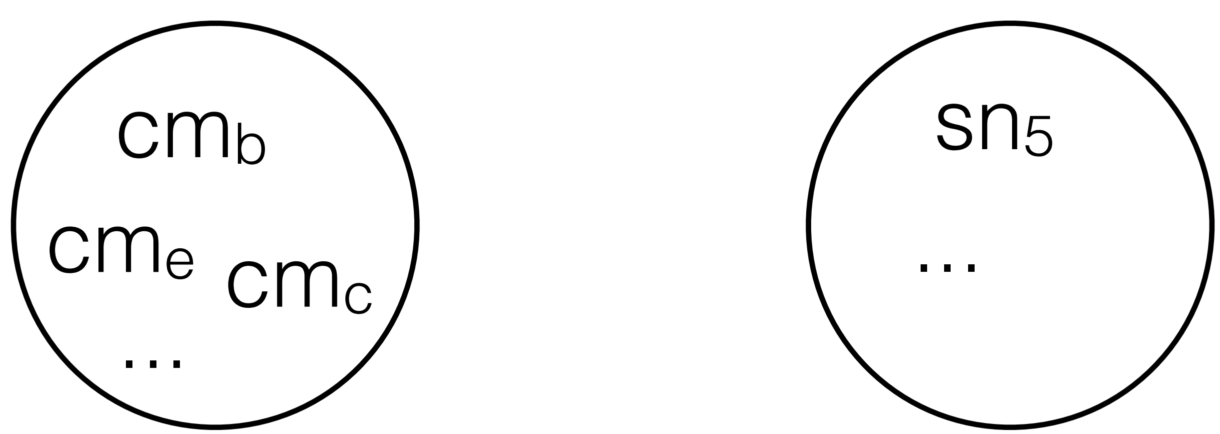
consumed coins
(serial numbers)

The Zerocash Paradigm [BCGGMTV14]

Each transaction consumes old coins and creates new coins.
The senders, receivers, and values are provably hidden.



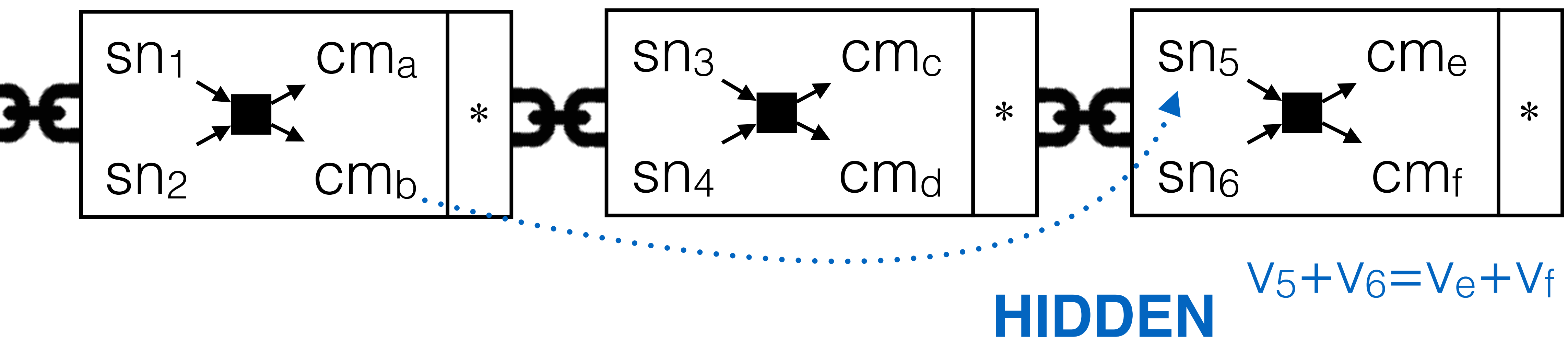
all created coins
(commitments)



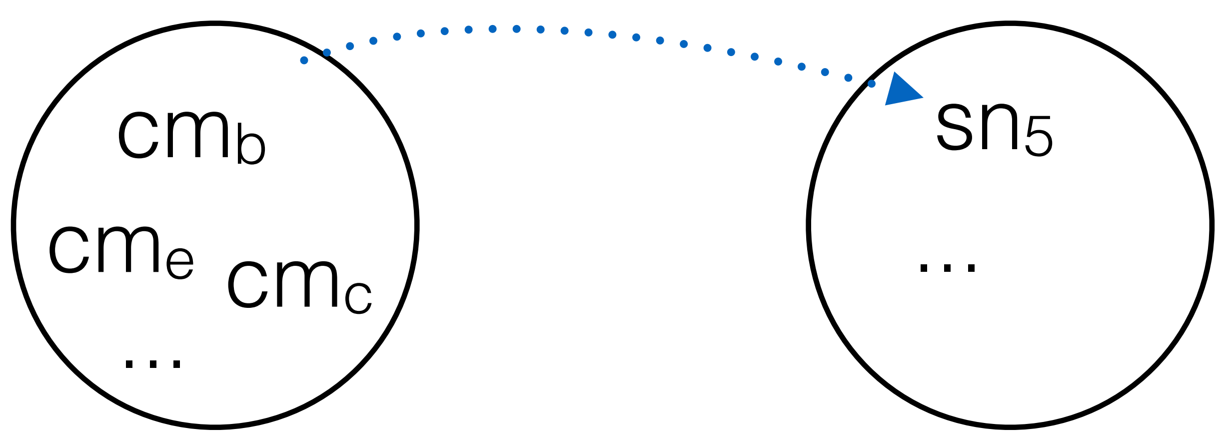
consumed coins
(serial numbers)

The Zerocash Paradigm [BCGGMTV14]

Each transaction consumes old coins and creates new coins.
The senders, receivers, and values are provably hidden.



all created coins
(commitments)



consumed coins
(serial numbers)

Sketch of Zerocash

[BCGGMTV14]

	old coin serial numbers	new coin commitments
tx	$\mathsf{sn}_i^{\mathsf{old}}$	$\mathsf{cm}_j^{\mathsf{new}}$

Simplify: just say that proof proves four things: Existence of coins, unique spends, construction of new coins, and correct values.
Reason: don't use it in the rest of the talk anyway.

Sketch of Zerocash

[BCGGMTV14]

	old coin serial numbers	new coin commitments	ZKP
tx	$\mathsf{sn}_i^{\mathsf{old}}$	$\mathsf{cm}_j^{\mathsf{new}}$	$\boldsymbol{\pi}$

Simplify: just say that proof proves four things: Existence of coins, unique spends, construction of new coins, and correct values.
Reason: don't use it in the rest of the talk anyway.

Sketch of Zerocash

[BCGGMTV14]

old coin serial numbers — new coin commitments — ZKP

tx | $\mathsf{sn}_i^{\mathsf{old}}$ | $\mathsf{cm}_j^{\mathsf{new}}$ | $\boldsymbol{\pi}$

Simplify: just say that proof proves four things: Existence of coins, unique spends, construction of new coins, and correct values.
Reason: don't use it in the rest of the talk anyway.

Sketch of Zerocash

[BCGGMTV14]

	set of all coin commitments	old coin serial numbers	new coin commitments	ZKP
tx	rt	$\mathsf{sn}_i^{\mathsf{old}}$	$\mathsf{cm}_j^{\mathsf{new}}$	$\boldsymbol{\pi}$

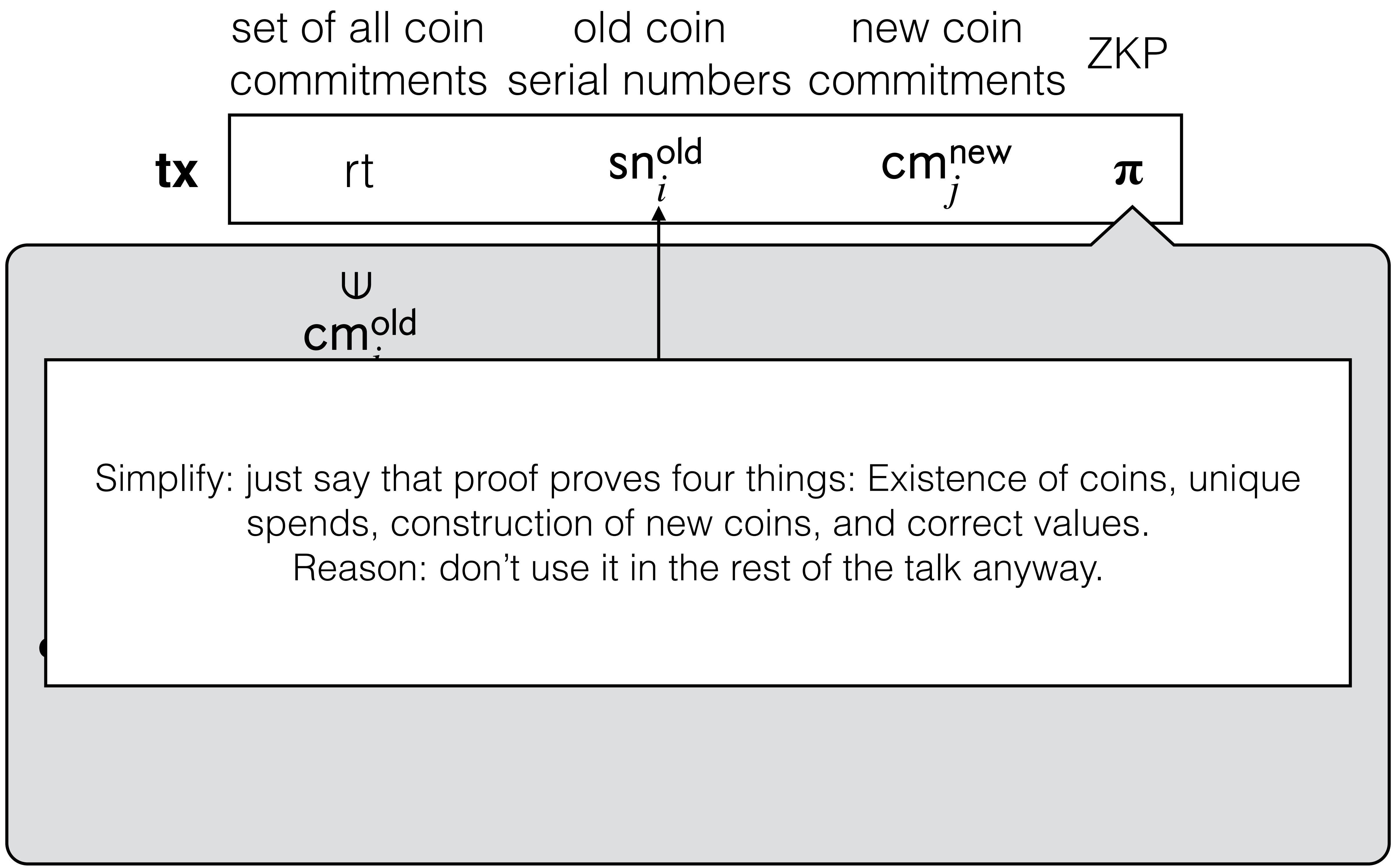
$\in$

$\mathsf{cm}_i^{\mathsf{old}}$

Simplify: just say that proof proves four things: Existence of coins, unique spends, construction of new coins, and correct values.
Reason: don't use it in the rest of the talk anyway.

Sketch of Zerocash

[BCGGMTV14]



Sketch of Zerocash

[BCGGMTV14]

	set of all coin commitments	old coin serial numbers	new coin commitments	ZKP
tx	rt	$\mathsf{sn}_i^{\mathsf{old}}$	$\mathsf{cm}_j^{\mathsf{new}}$	$\boldsymbol{\pi}$

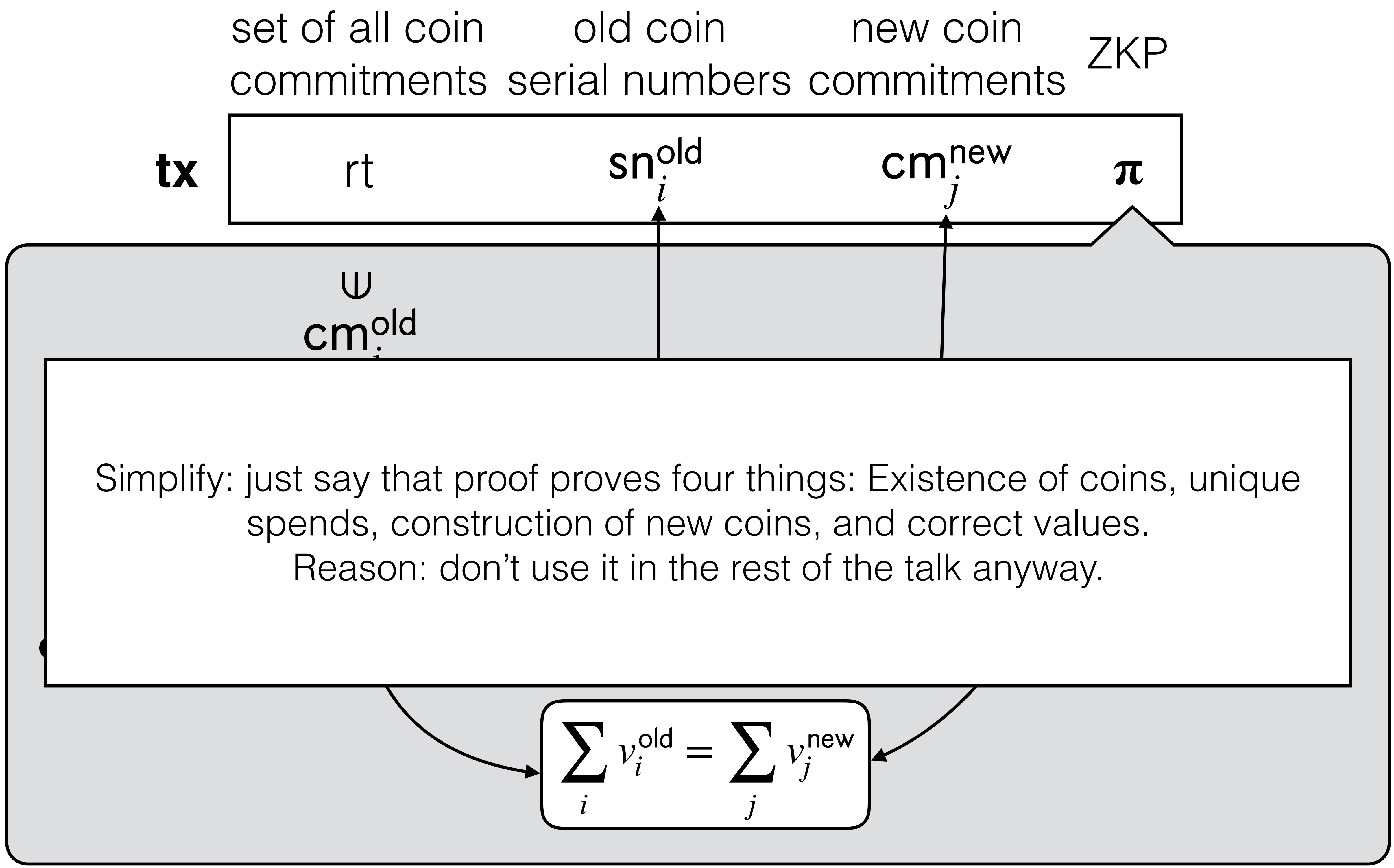
$\Cup$

$\mathsf{cm}_i^{\mathsf{old}}$

Simplify: just say that proof proves four things: Existence of coins, unique spends, construction of new coins, and correct values.
Reason: don’t use it in the rest of the talk anyway.

Sketch of Zerocash

[BCGGMTV14]

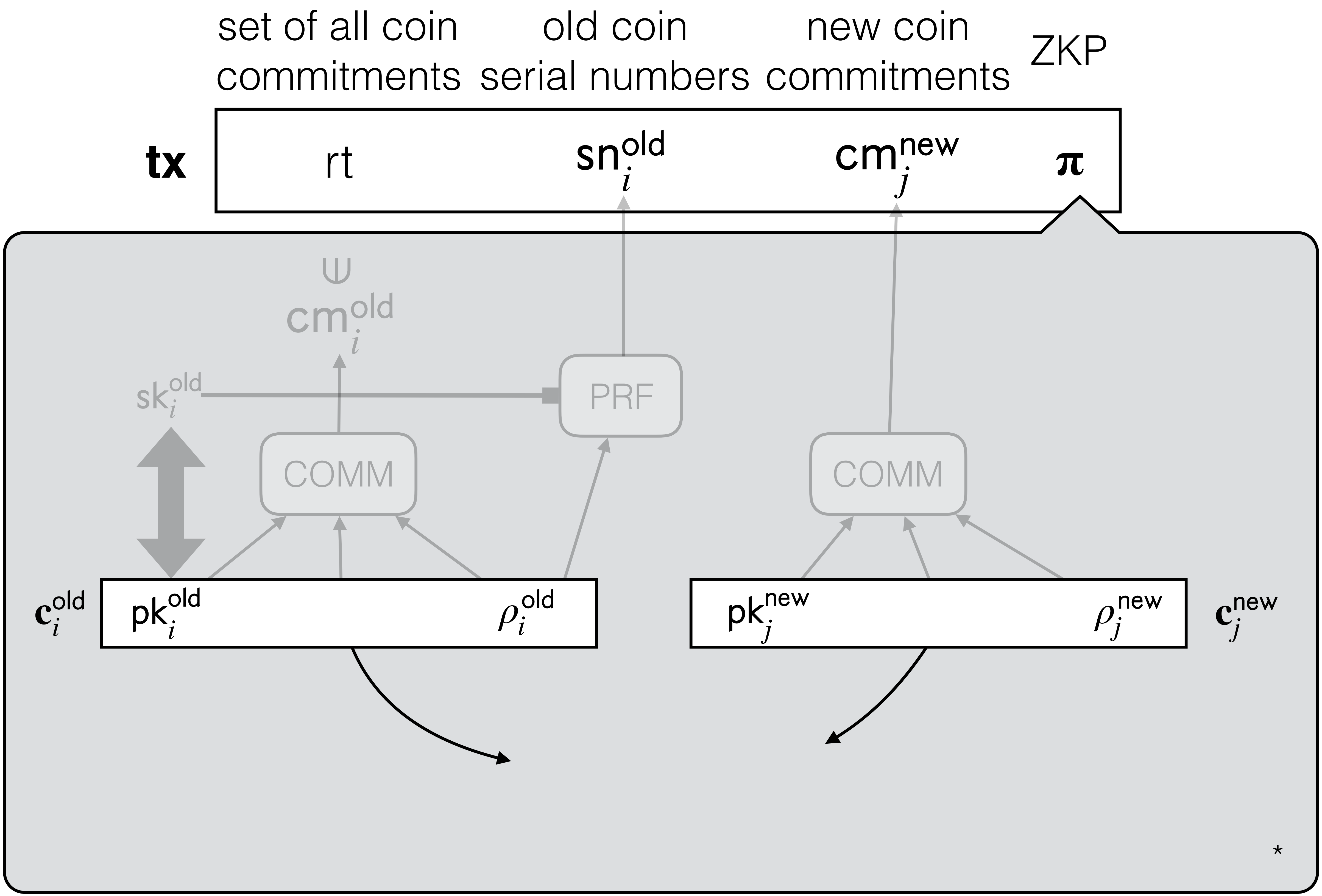


Zerocash achieves ideal anonymity for a single asset (on a single blockchain).

Zerocash achieves ideal anonymity for a single asset (on a single blockchain).

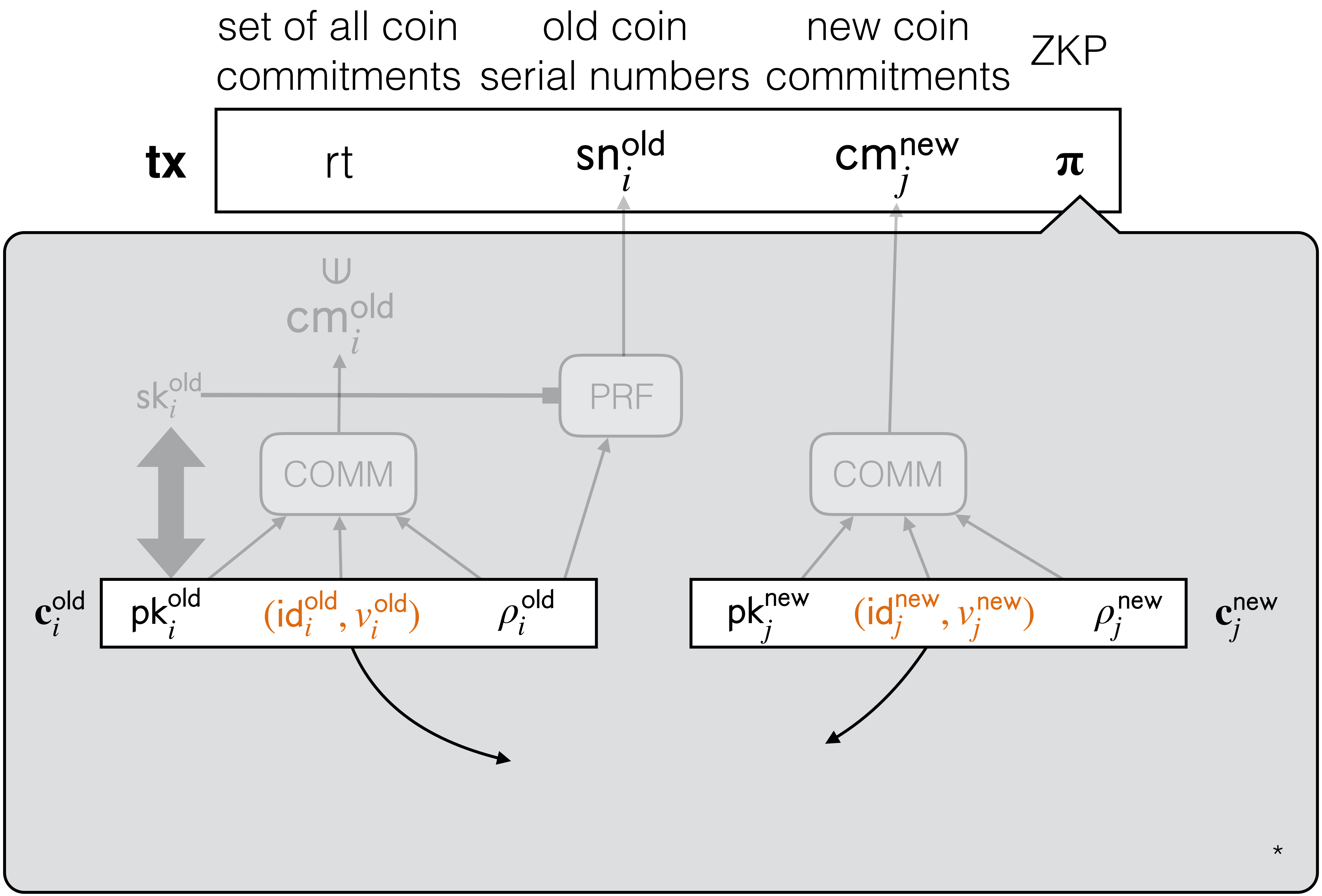
How to achieve ideal privacy for multiple user-defined assets (on the same blockchain)?

Private user-defined assets



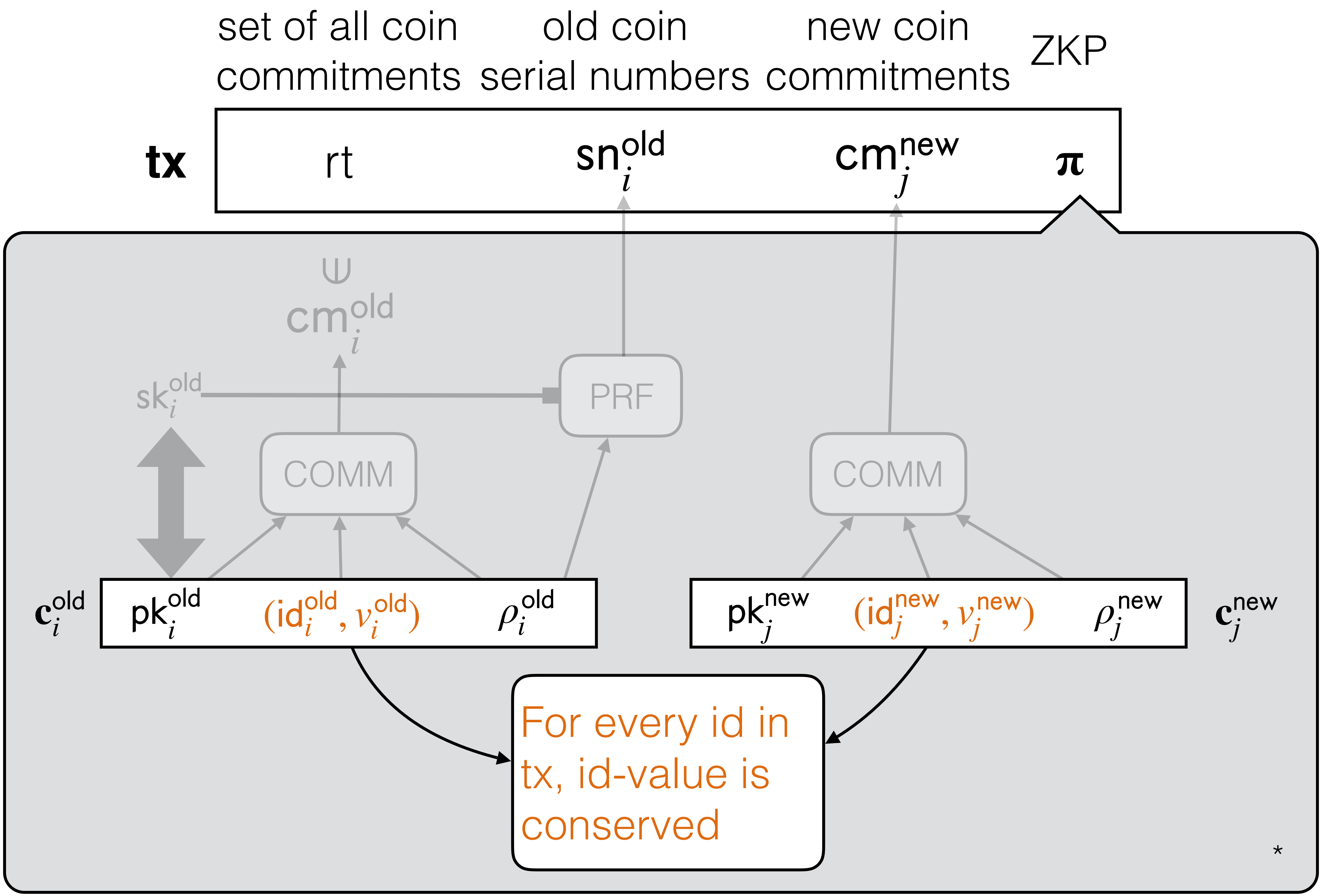
* For now we ignore how to mint an initial supply and generate a unique id.

Private user-defined assets



* For now we ignore how to mint an initial supply and generate a unique id.

Private user-defined assets



* For now we ignore how to mint an initial supply and generate a unique id.

So far: multiple, private, user-defined assets in the same transaction, with the same anonymity pool.

So far: multiple, private, user-defined assets in the same transaction, with the same anonymity pool.

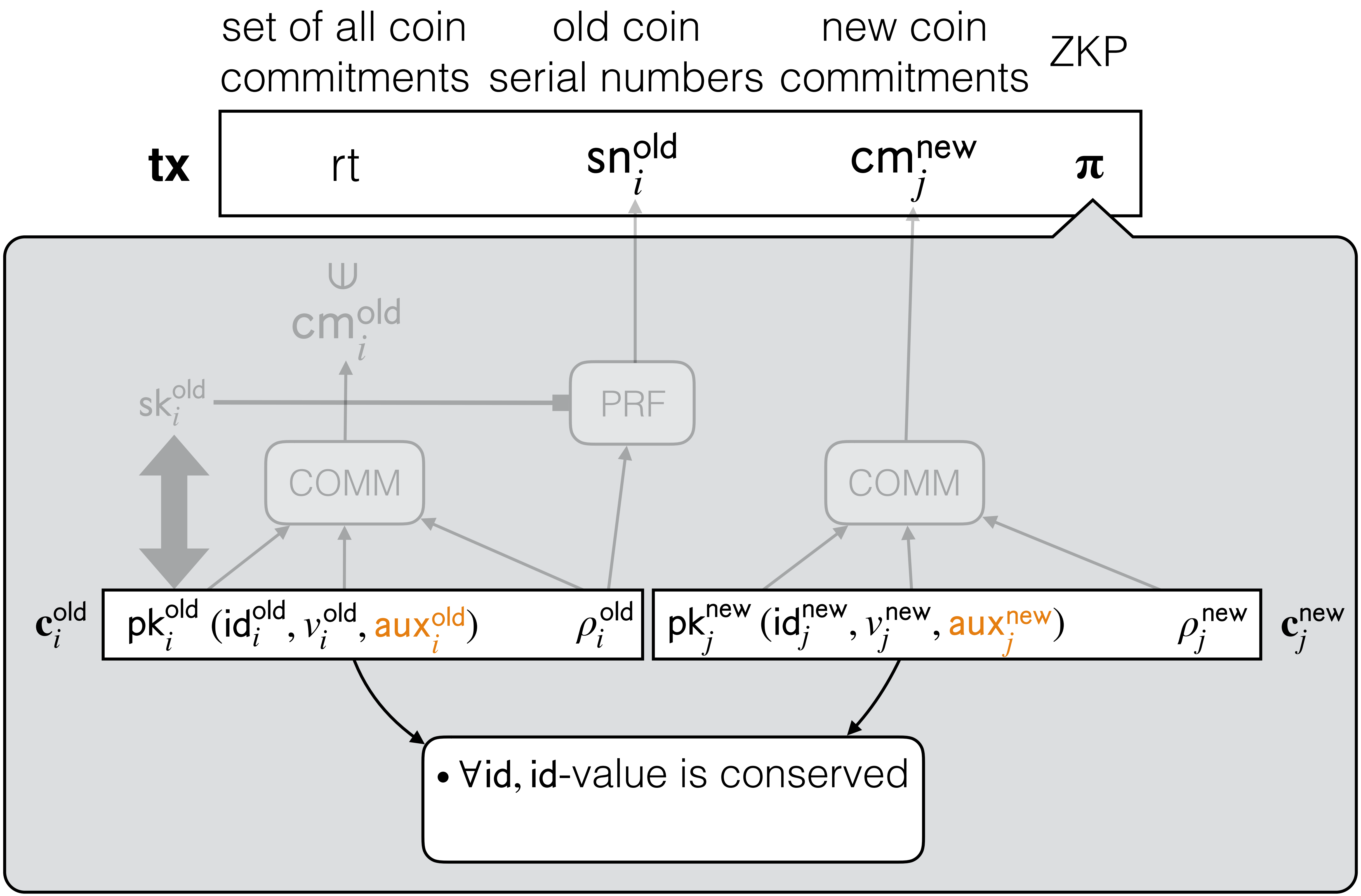
But: assets are still isolated.

So far: multiple, private, user-defined assets in the same transaction, with the same anonymity pool.

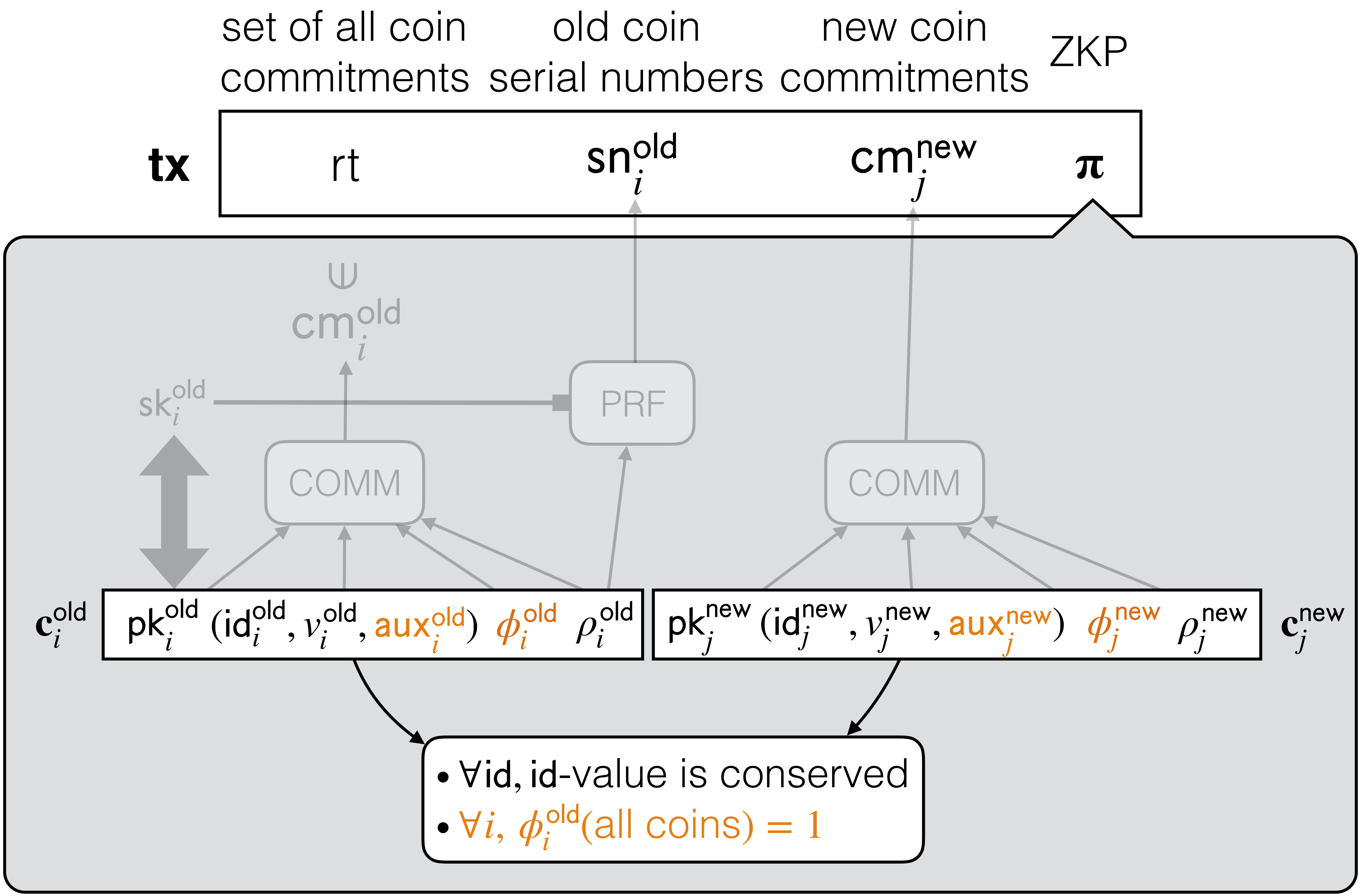
But: assets are still isolated.

How to enable applications that interact with multiple assets?

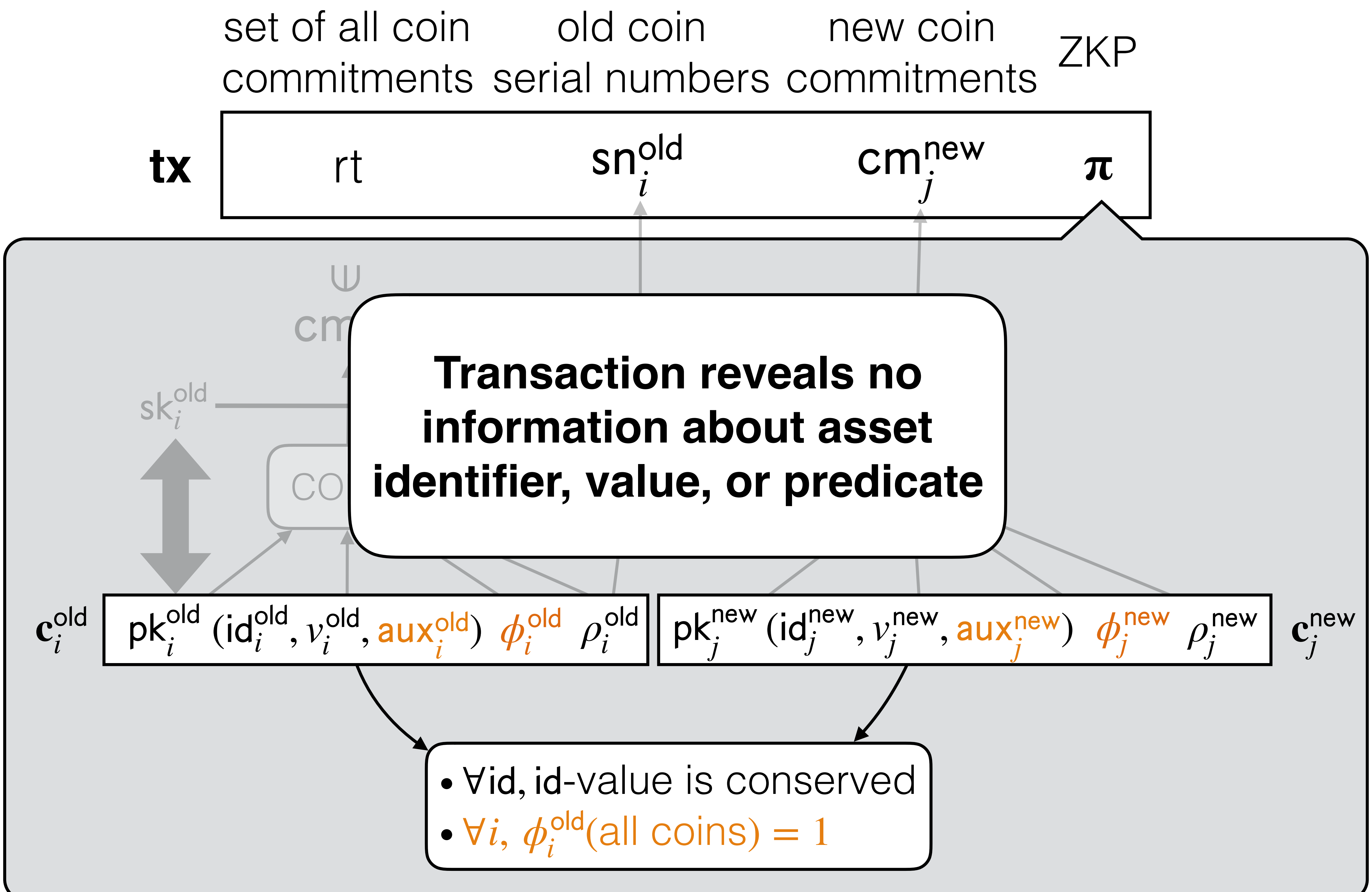
Custom access to user-defined assets



Custom access to user-defined assets



Custom access to user-defined assets



Private atomic swaps

Private atomic swaps

Key primitive for constructing DEXs

Private atomic swaps

Key primitive for constructing DEXs

$\mathbf{c}_1^{\mathsf{old}}$

$\mathsf{pk}_1^{\mathsf{old}}$

$(\mathsf{id}_1^{\mathsf{old}}, v_1^{\mathsf{old}}, \mathsf{aux}_1^{\mathsf{old}})$

$\phi_1^{\mathsf{old}} = \phi_{\mathsf{exch}}$

Private atomic swaps

Key primitive for constructing DEXs

$\mathbf{c}_1^{\text{old}}$

$\mathsf{pk}_1^{\text{old}}$

$(\mathsf{id}_1^{\text{old}}, v_1^{\text{old}}, \mathsf{aux}_1^{\text{old}})$

$\phi_1^{\text{old}} = \phi_{\text{exch}}$

$\mathbf{c}_2^{\text{old}}$

$\mathsf{pk}_2^{\text{old}}$

$(\mathsf{id}_2^{\text{old}}, v_2^{\text{old}}, \mathsf{aux}_2^{\text{old}})$

ϕ_2^{old}

$\mathbf{c}_1^{\text{new}}$

$\mathsf{pk}_1^{\text{new}}$

$(\mathsf{id}_1^{\text{new}}, v_1^{\text{new}}, \perp)$

ϕ_1^{new}

$\mathbf{c}_2^{\text{new}}$

$\mathsf{pk}_2^{\text{new}}$

$(\mathsf{id}_2^{\text{new}}, v_2^{\text{new}}, \perp)$

ϕ_2^{new}

Private atomic swaps

Key primitive for constructing DEXs

$\mathbf{c}_1^{\mathsf{old}}$

$\mathsf{pk}_1^{\mathsf{old}}$

$(\mathsf{id}_1^{\mathsf{old}}, v_1^{\mathsf{old}}, \mathsf{aux}_1^{\mathsf{old}})$

$\phi_1^{\mathsf{old}} = \phi_{\mathsf{exch}}$

$\mathbf{c}_2^{\mathsf{old}}$

$\mathsf{pk}_2^{\mathsf{old}}$

$(\mathsf{id}_2^{\mathsf{old}}, v_2^{\mathsf{old}}, \mathsf{aux}_2^{\mathsf{old}})$

ϕ_2^{old}

$\mathbf{c}_1^{\mathsf{new}}$

$\mathsf{pk}_1^{\mathsf{new}}$

$(\mathsf{id}_1^{\mathsf{new}}, v_1^{\mathsf{new}}, \perp)$

ϕ_1^{new}

$\mathbf{c}_2^{\mathsf{new}}$

$\mathsf{pk}_2^{\mathsf{new}}$

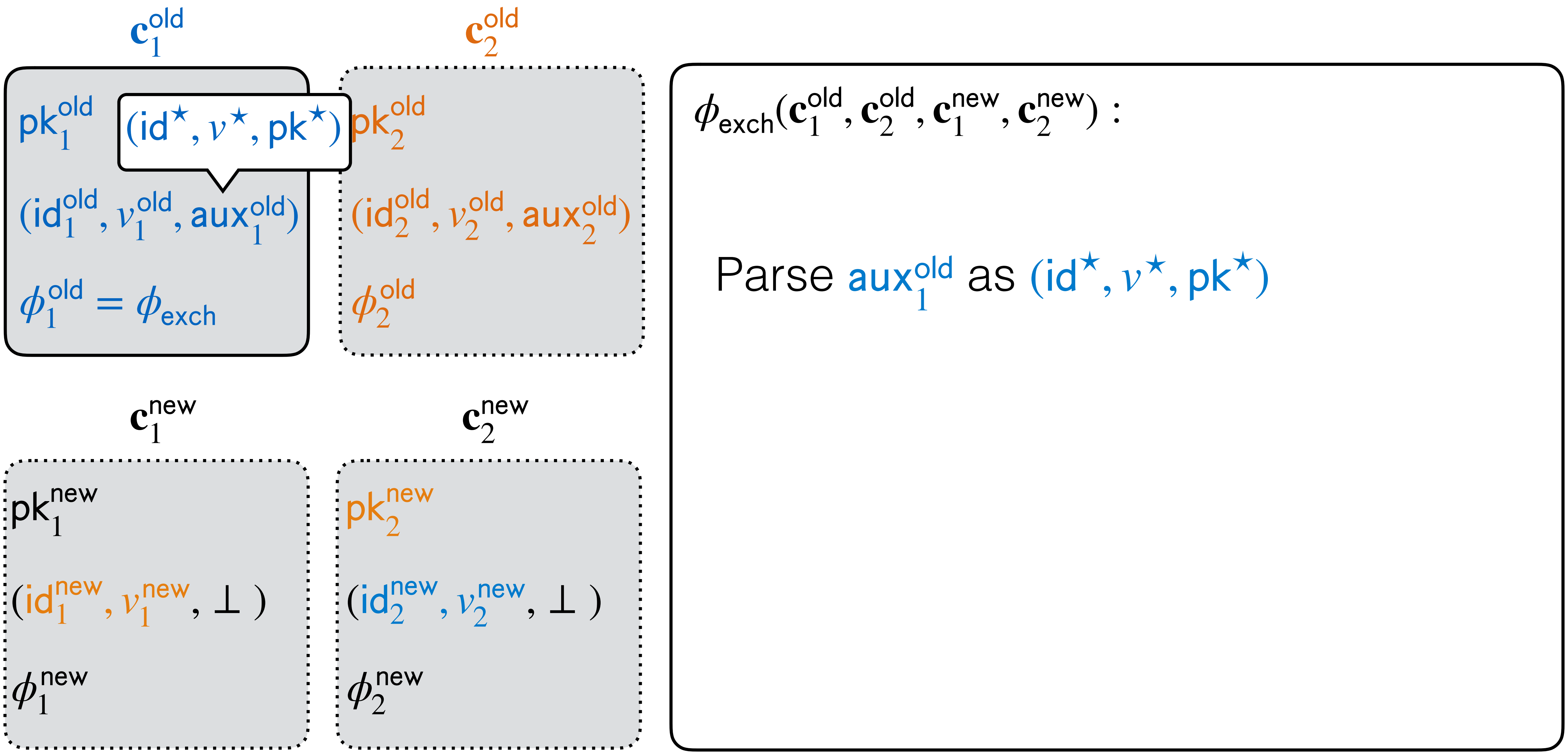
$(\mathsf{id}_2^{\mathsf{new}}, v_2^{\mathsf{new}}, \perp)$

ϕ_2^{new}

$\phi_{\mathsf{exch}}(\mathbf{c}_1^{\mathsf{old}}, \mathbf{c}_2^{\mathsf{old}}, \mathbf{c}_1^{\mathsf{new}}, \mathbf{c}_2^{\mathsf{new}})$:

Private atomic swaps

Key primitive for constructing DEXs



Private atomic swaps

Key primitive for constructing DEXs

$\mathbf{c}_1^{\text{old}}$

pk_1^{old}

$(\text{id}^\star, v^\star, \text{pk}^\star)$

$(\text{id}_1^{\text{old}}, v_1^{\text{old}}, \text{aux}_1^{\text{old}})$

$\phi_1^{\text{old}} = \phi_{\text{exch}}$

$\mathbf{c}_2^{\text{old}}$

pk_2^{old}

$(\text{id}_2^{\text{old}}, v_2^{\text{old}}, \text{aux}_2^{\text{old}})$

ϕ_2^{old}

$\mathbf{c}_1^{\text{new}}$

pk_1^{new}

$(\text{id}_1^{\text{new}}, v_1^{\text{new}}, \perp)$

ϕ_1^{new}

$\mathbf{c}_2^{\text{new}}$

pk_2^{new}

$(\text{id}_2^{\text{new}}, v_2^{\text{new}}, \perp)$

ϕ_2^{new}

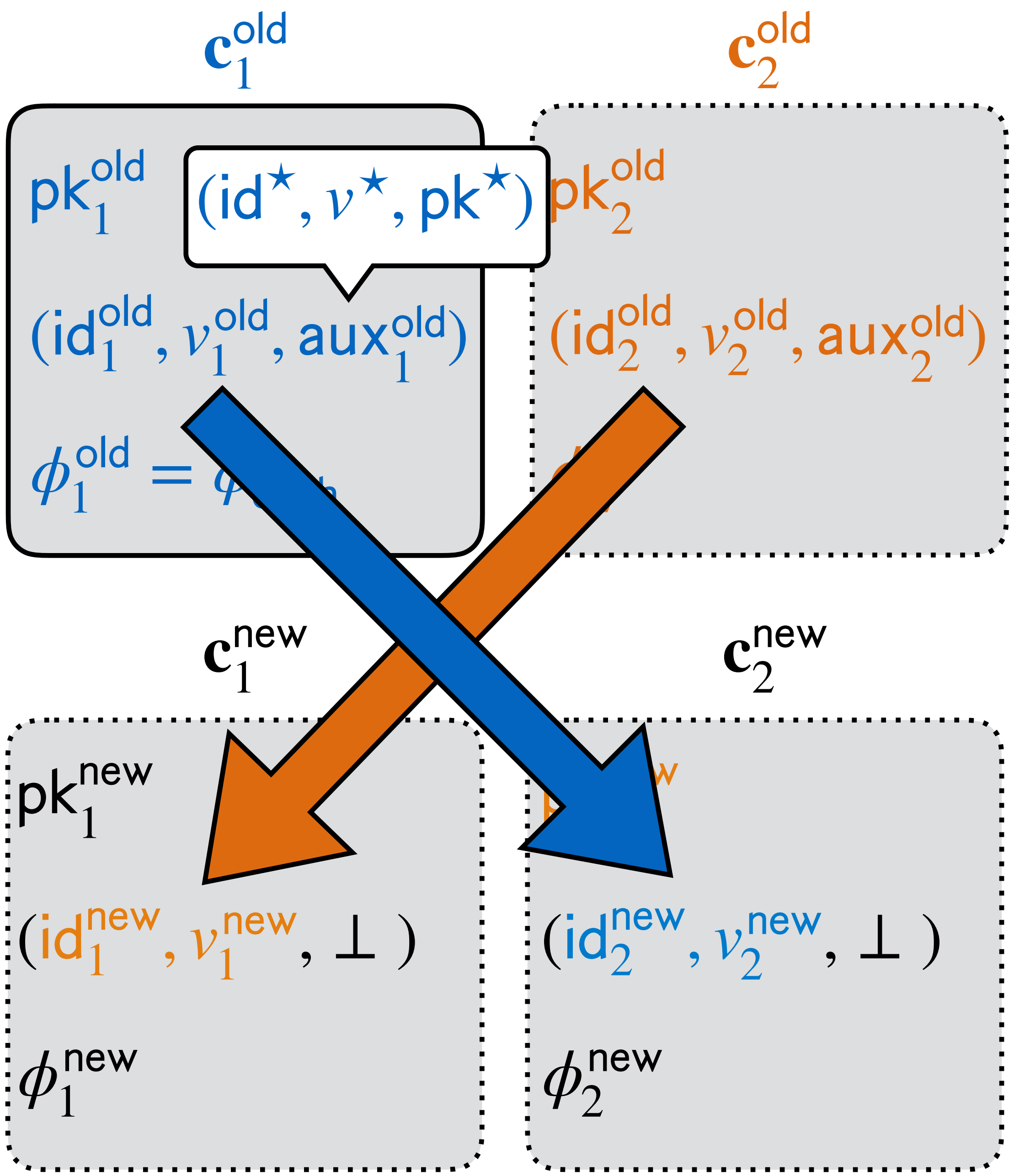
$\phi_{\text{exch}}(\mathbf{c}_1^{\text{old}}, \mathbf{c}_2^{\text{old}}, \mathbf{c}_1^{\text{new}}, \mathbf{c}_2^{\text{new}})$:

Parse $\text{aux}_1^{\text{old}}$ as $(\text{id}^\star, v^\star, \text{pk}^\star)$

Check that $\text{id}_2^{\text{old}} = \text{id}^\star$ and $v_2^{\text{old}} = v^\star$

Private atomic swaps

Key primitive for constructing DEXs



$\phi_{\text{exch}}(\mathbf{c}_1^{\text{old}}, \mathbf{c}_2^{\text{old}}, \mathbf{c}_1^{\text{new}}, \mathbf{c}_2^{\text{new}}):$

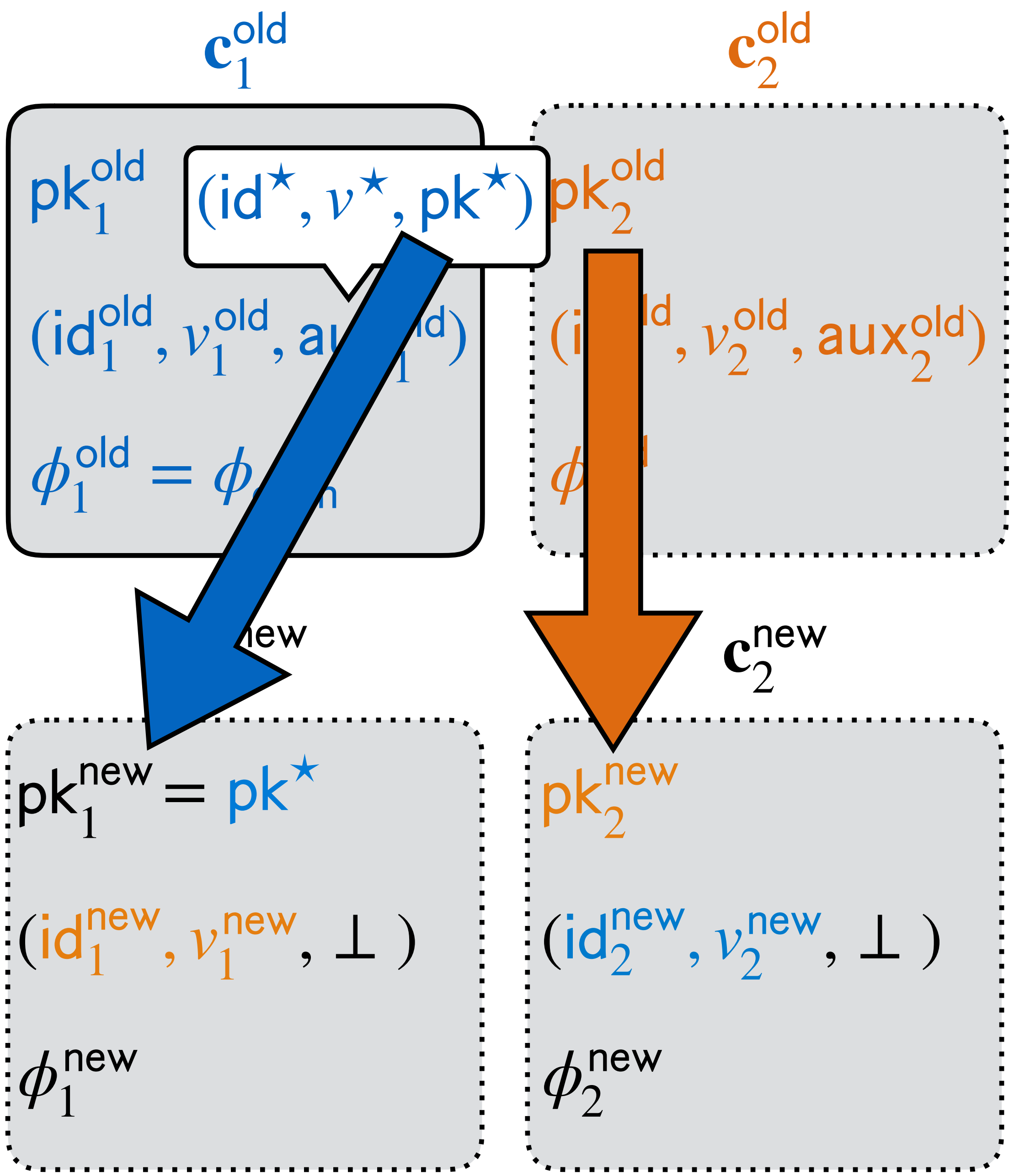
Parse $\text{aux}_1^{\text{old}}$ as $(\text{id}^\star, v^\star, \text{pk}^\star)$

Check that $\text{id}_2^{\text{old}} = \text{id}^\star$ and $v_2^{\text{old}} = v^\star$

Perform swap: $\text{id}_1^{\text{new}} = \text{id}_2^{\text{old}}; \quad v_1^{\text{new}} = v_2^{\text{old}}$
$\text{id}_2^{\text{new}} = \text{id}_1^{\text{old}}; \quad v_2^{\text{new}} = v_1^{\text{old}}$

Private atomic swaps

Key primitive for constructing DEXs



$\phi_{\text{exch}}(\mathbf{c}_1^{\text{old}}, \mathbf{c}_2^{\text{old}}, \mathbf{c}_1^{\text{new}}, \mathbf{c}_2^{\text{new}})$:

Parse $\mathsf{aux}_1^{\text{old}}$ as $(\mathsf{id}^\star, v^\star, \mathsf{pk}^\star)$

Check that $\mathsf{id}_2^{\text{old}} = \mathsf{id}^\star$ and $v_2^{\text{old}} = v^\star$

Perform swap: $\mathsf{id}_1^{\text{new}} = \mathsf{id}_2^{\text{old}}; \quad v_1^{\text{new}} = v_2^{\text{old}}$
$\mathsf{id}_2^{\text{new}} = \mathsf{id}_1^{\text{old}}; \quad v_2^{\text{new}} = v_1^{\text{old}}$

Check addresses: $\mathsf{pk}_1^{\text{new}} = \mathsf{pk}^\star$
$\mathsf{pk}_2^{\text{new}} = \mathsf{pk}_2^{\text{old}}$

DEX

Atomic swap ✓

DEX

DEX

- Atomic swap ✓
- Order creation
- Order discovery
- Trade finalization

DEX Architectures

DEX Architectures

Order-based:

DEX Architectures

Order-based:

- *Order book* maintains list of orders published by makers

DEX Architectures

Order-based:

$(A, B, v_A, v_B, \ldots)$

- *Order book* maintains list of orders published by makers

DEX Architectures

Order-based:

$(A, B, v_A, v_B, \ldots)$

- *Order book* maintains list of orders published by makers
- Takers can scan for open orders and fill them

DEX Architectures

Order-based:

$(A, B, v_A, v_B, \ldots)$

- *Order book* maintains list of orders published by makers
- Takers can scan for open orders and fill them
- Eg: Radar Relay, IDEX.

DEX Architectures

Order-based:

$(A, B, v_A, v_B, \ldots)$

- *Order book* maintains list of orders published by makers
- Takers can scan for open orders and fill them
- Eg: Radar Relay, IDEX.

Index-based:

DEX Architectures

Order-based:

$(A, B, v_A, v_B, \ldots)$

- *Order book* maintains list of orders published by makers
- Takers can scan for open orders and fill them
- Eg: Radar Relay, IDEX.

Index-based:

- *Index* maintains list of “intents-to-trade” published by makers

DEX Architectures

Order-based:

$(A, B, v_A, v_B, \ldots)$

- *Order book* maintains list of orders published by makers
- Takers can scan for open orders and fill them
- Eg: Radar Relay, IDEX.

Index-based:

$(A, B, \ldots)$

- *Index* maintains list of “intents-to-trade” published by makers

DEX Architectures

Order-based:

$(A, B, v_A, v_B, \ldots)$

- *Order book* maintains list of orders published by makers
- Takers can scan for open orders and fill them
- Eg: Radar Relay, IDEX.

Index-based:

$(A, B, \ldots)$

- *Index* maintains list of “intents-to-trade” published by makers
- Takers can scan for intentions and interact with makers to fill orders

DEX Architectures

Order-based:

$(A, B, v_A, v_B, \ldots)$

- *Order book* maintains list of orders published by makers
- Takers can scan for open orders and fill them
- Eg: Radar Relay, IDEX.

Index-based:

$(A, B, \ldots)$

- *Index* maintains list of “intents-to-trade” published by makers
- Takers can scan for intentions and interact with makers to fill orders
- Eg: AirSwap

DEX Architectures

Order-based:

$(A, B, v_A, v_B, \ldots)$

- *Order book* maintains list of orders published by makers
- Takers can scan for open orders and fill them
- Eg: Radar Relay, IDEX.

Index-based:

$(A, B, \ldots)$

- *Index* maintains list of “intents-to-trade” published by makers
- Takers can scan for intentions and interact with makers to fill orders
- Eg: AirSwap

Intent-based DEX

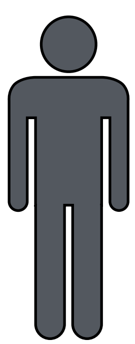
Index

Sell	Buy	Public key
BAT	MKR	pk_1
ZRX	OMG	pk_2

Intent-based DEX

Index

Sell	Buy	Public key
BAT	MKR	pk_1
ZRX	OMG	pk_2

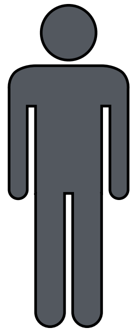


Maker

Intent-based DEX

Index

Sell	**Buy**	**Public key**
BAT	MKR	pk_1
ZRX	OMG	pk_2
A	B	pk_M



Maker

Intent-based DEX

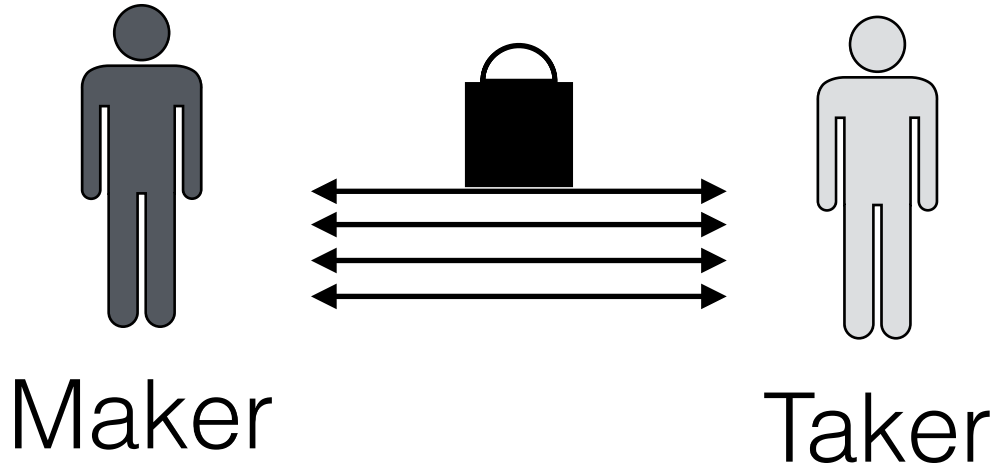
Index

Sell	**Buy**	**Public key**
BAT	MKR	pk_1
ZRX	OMG	pk_2
A	B	pk_M

Intent-based DEX

Index

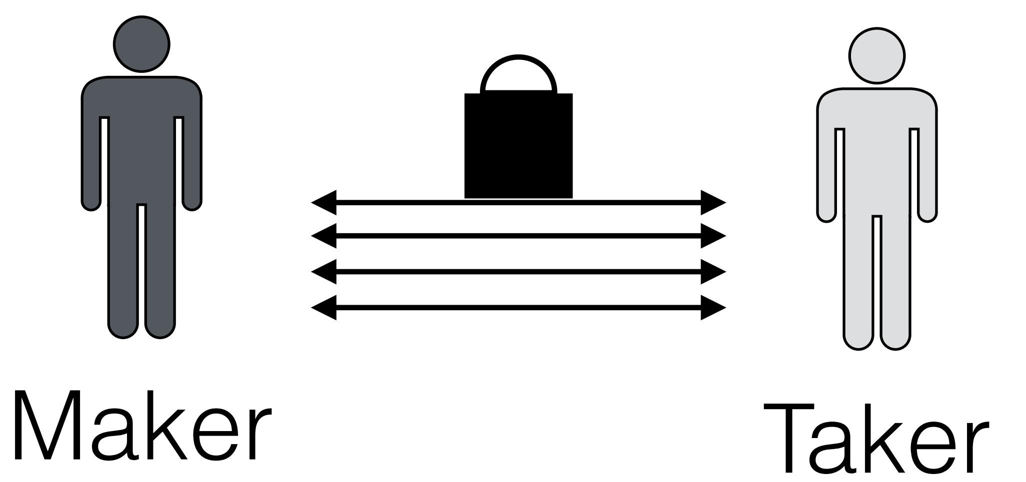
Sell	Buy	Public key
BAT	MKR	pk_1
ZRX	OMG	pk_2
A	B	pk_M



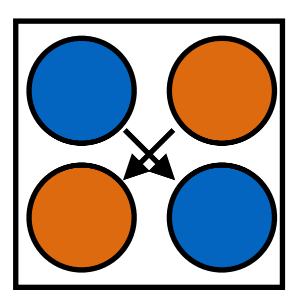
Intent-based DEX

Index

Sell	Buy	Public key
BAT	MKR	pk_1
ZRX	OMG	pk_2
A	B	pk_M



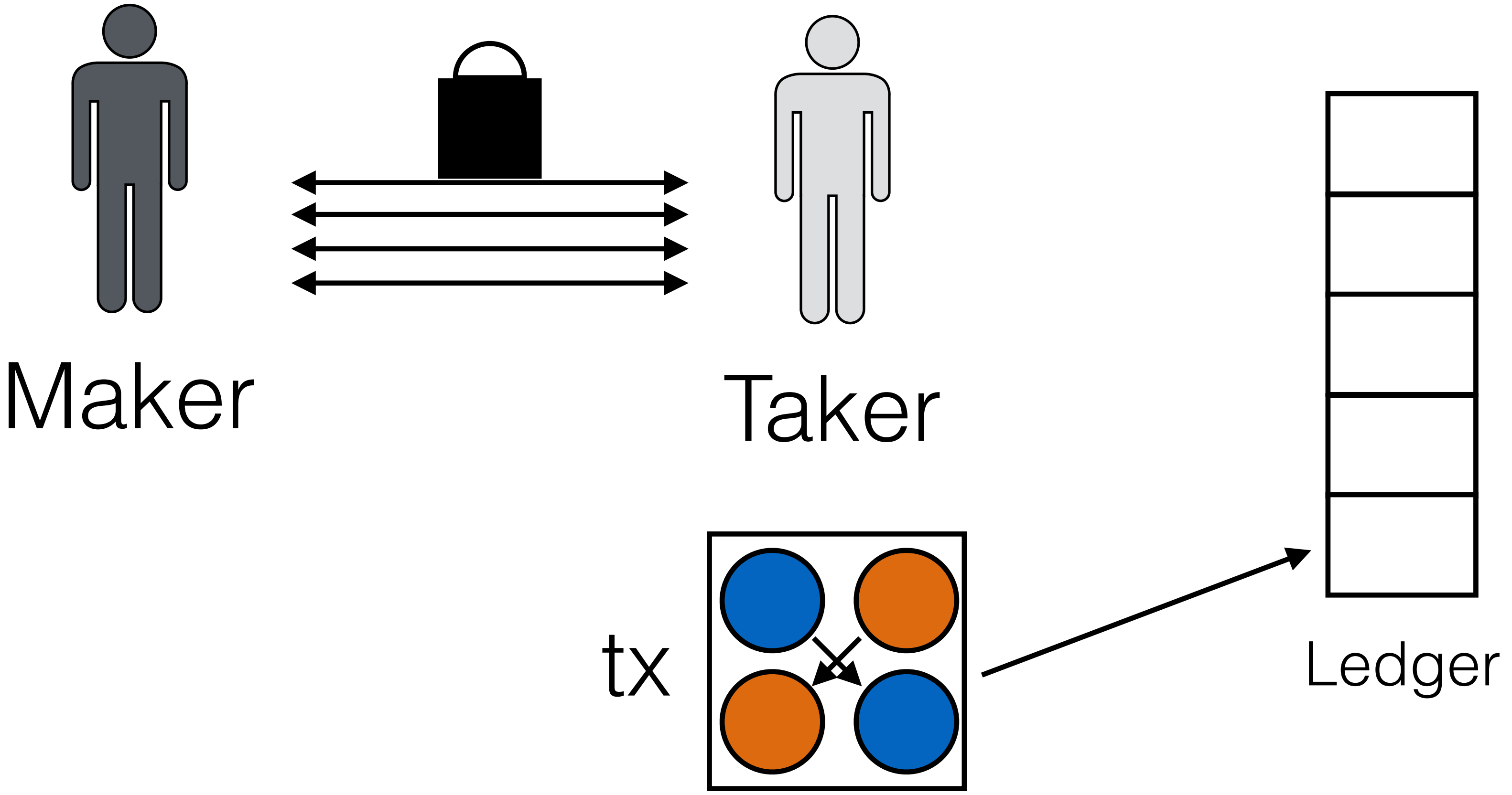
tx



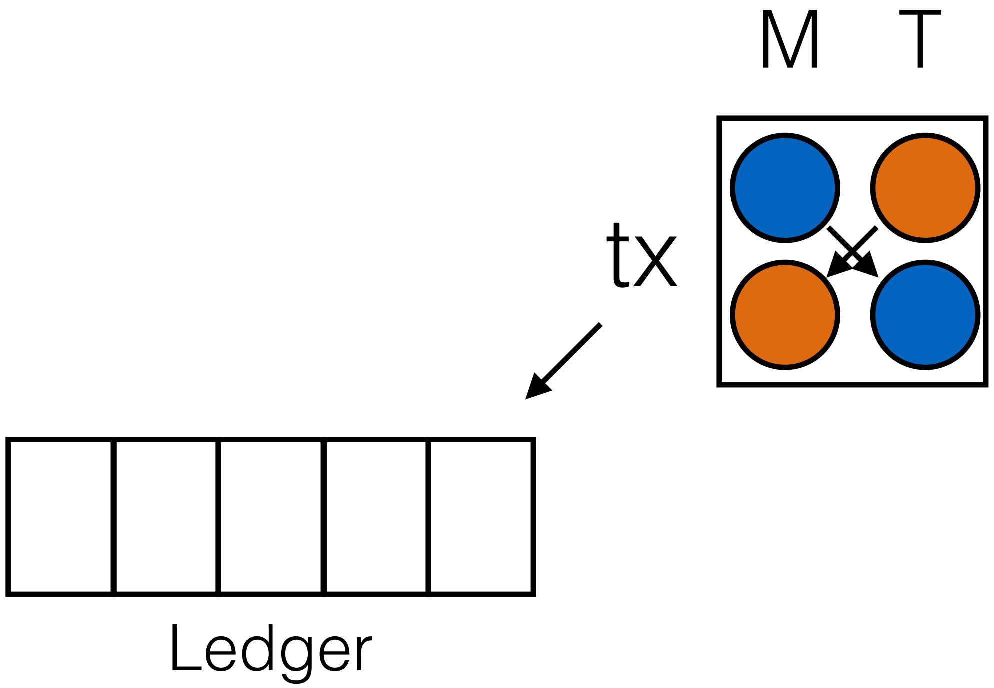
Intent-based DEX

Index

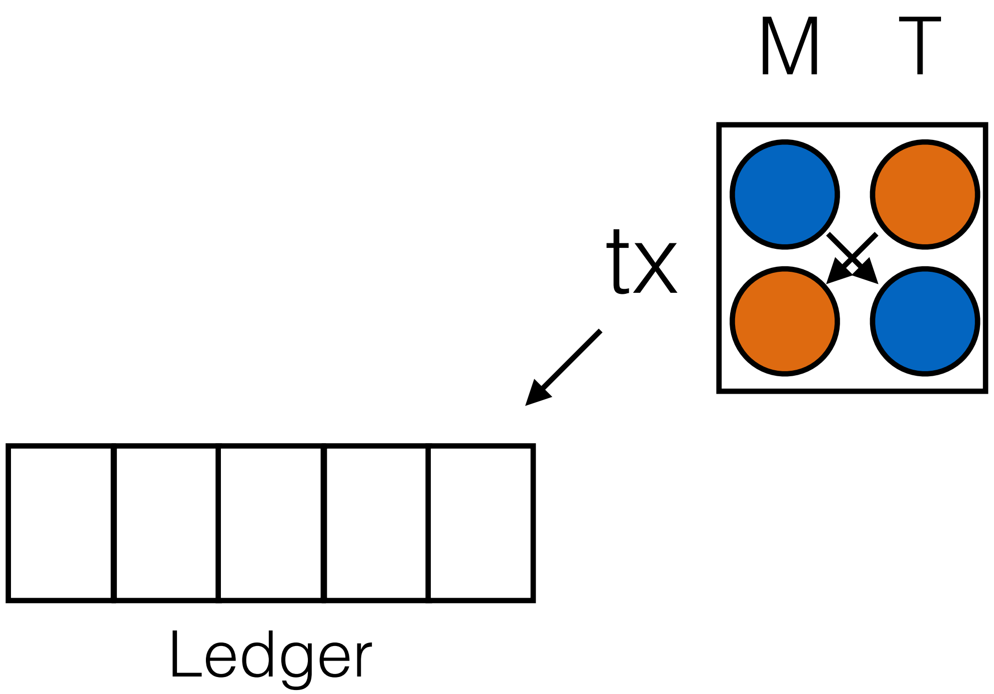
Sell	Buy	Public key
BAT	MKR	pk_1
ZRX	OMG	pk_2
A	B	pk_M



Privacy Leakage



Privacy Leakage



“The addresses

$addr_M$ and $addr_T$

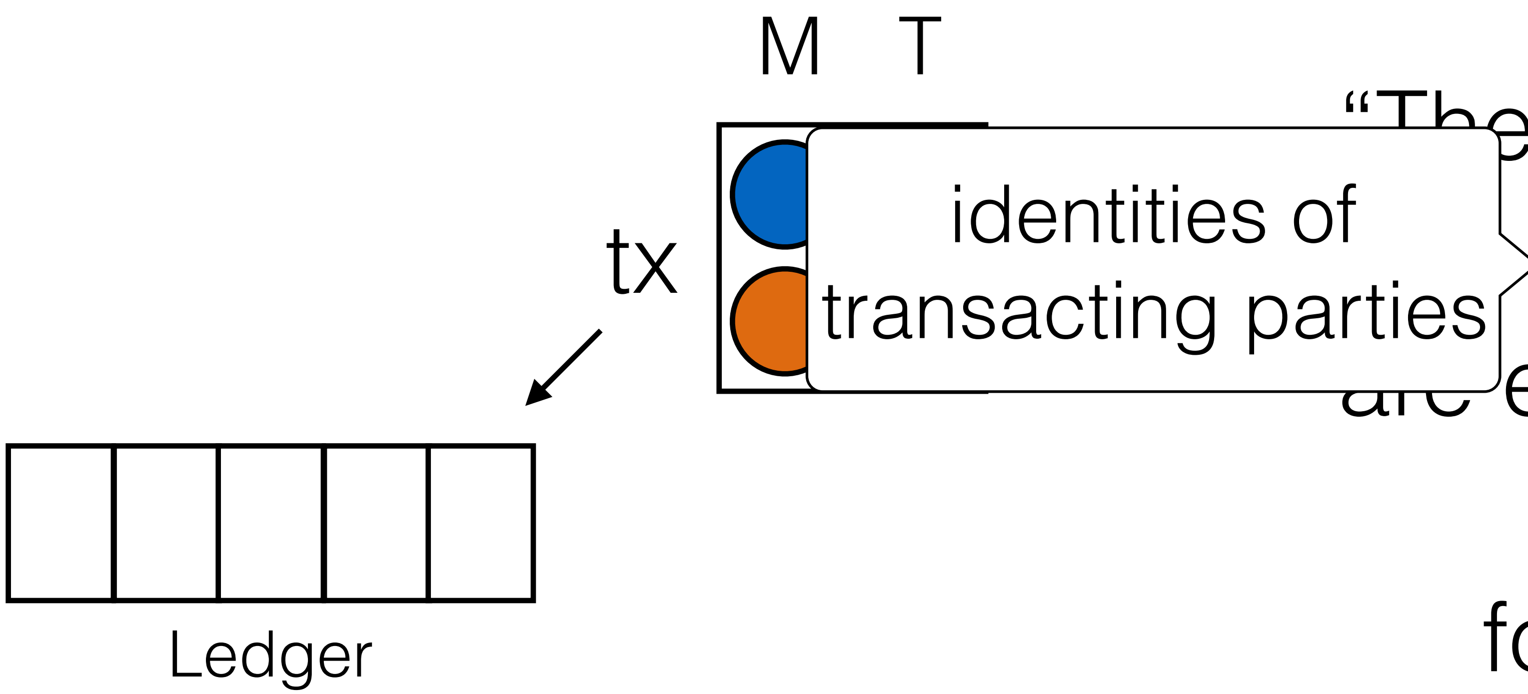
are exchanging

v_A units of A

for

v_B units of B”

Privacy Leakage



“The addresses

$addr_M$ and $addr_T$

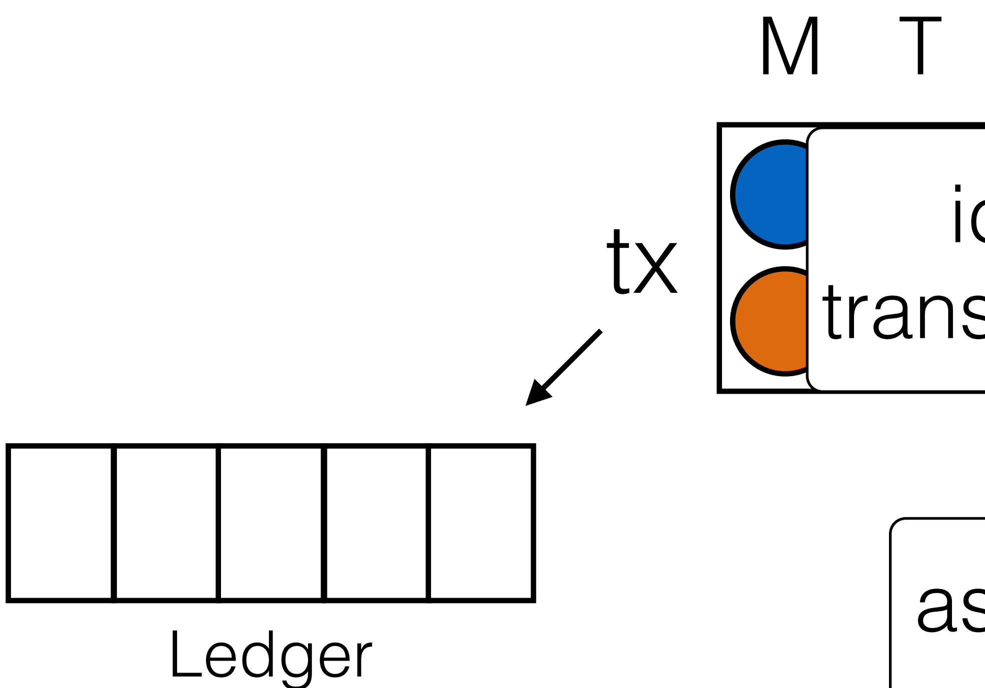
are exchanging

v_A units of A

for

v_B units of B”

Privacy Leakage



identities of transacting parties

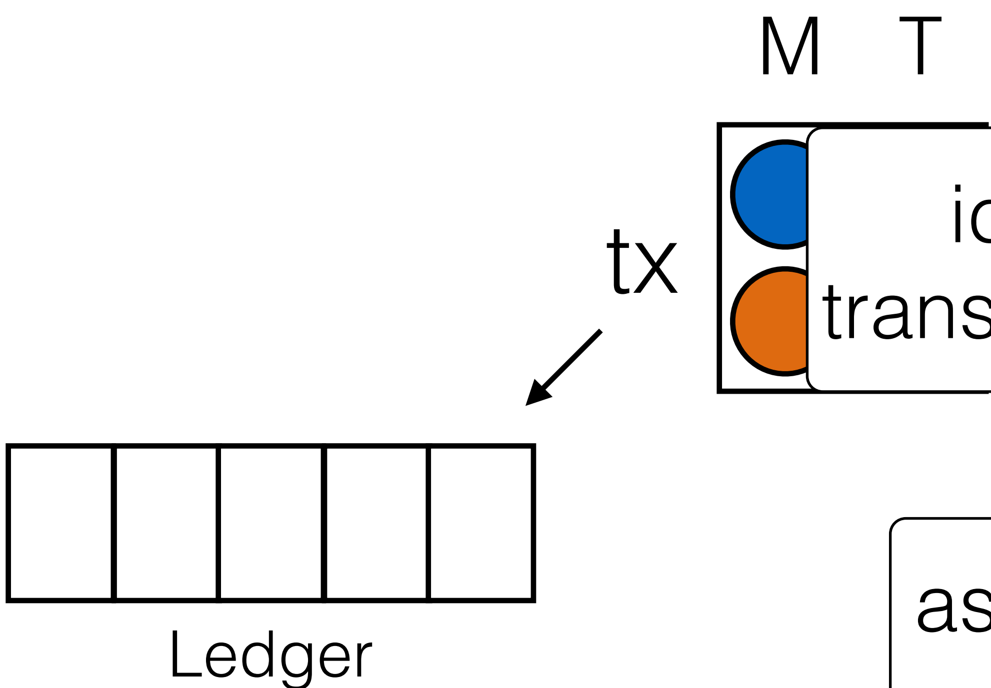
assets and values in the trade

“The addresses **$addr_M$ and $addr_T$** are exchanging

v_A units of A

v_B units of B”

Privacy Leakage



“The addresses **$addr_M$ and $addr_T$** are exchanging

identities of transacting parties

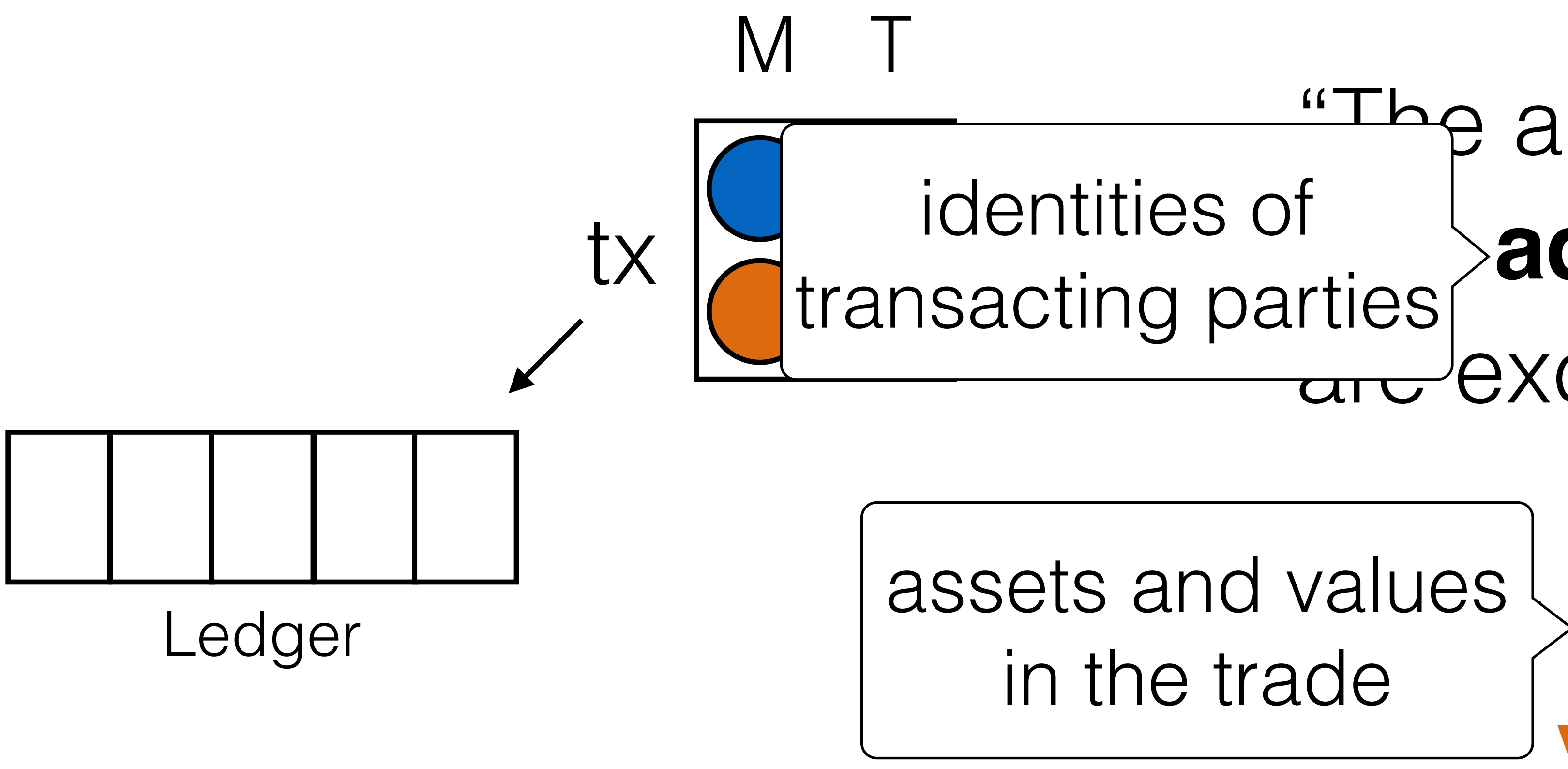
v_A units of A

assets and values in the trade

v_B units of B”

Trade anonymity
No information about participants in the trade is revealed.

Privacy Leakage



“The addresses **$addr_M$ and $addr_T$** are exchanging **v_A units of A** … **v_B units of B**”

Trade anonymity
No information about participants in the trade is revealed.

Trade confidentiality
No information about the assets and values used in the trade is revealed.

Private Intent-based DEX

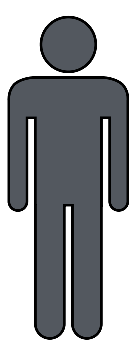
Index

Sell	Buy	Public key
BAT	MKR	pk_1
ZRX	OMG	pk_2

Private Intent-based DEX

Index

Sell	Buy	Public key
BAT	MKR	pk_1
ZRX	OMG	pk_2

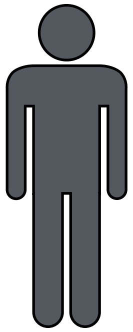


Maker

Private Intent-based DEX

Index

Sell	Buy	Public key
BAT	MKR	pk_1
ZRX	OMG	pk_2
A	B	pk_M

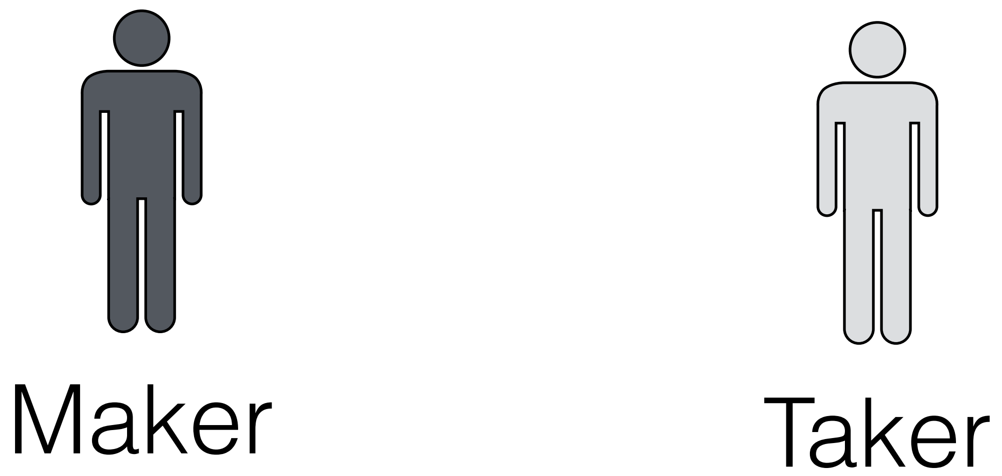


Maker

Private Intent-based DEX

Index

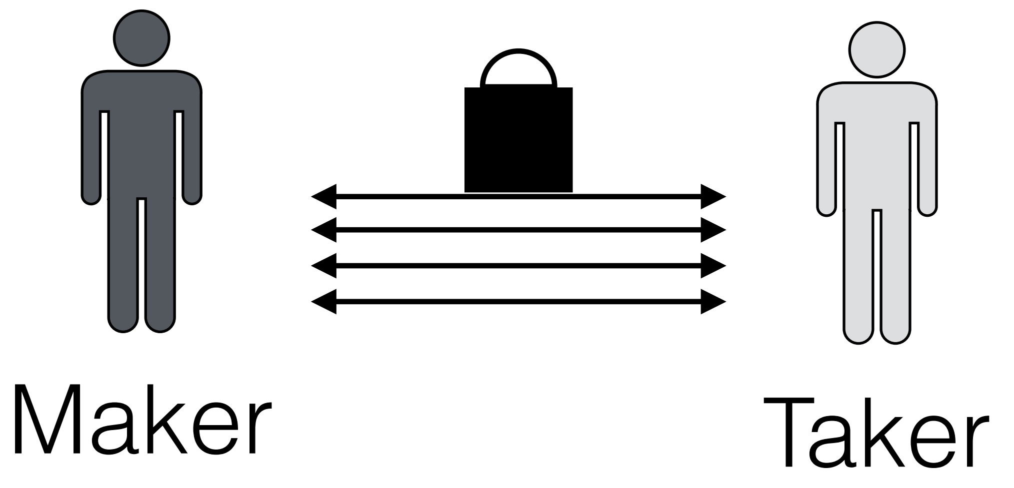
Sell	Buy	Public key
BAT	MKR	pk_1
ZRX	OMG	pk_2
A	B	pk_M



Private Intent-based DEX

Index

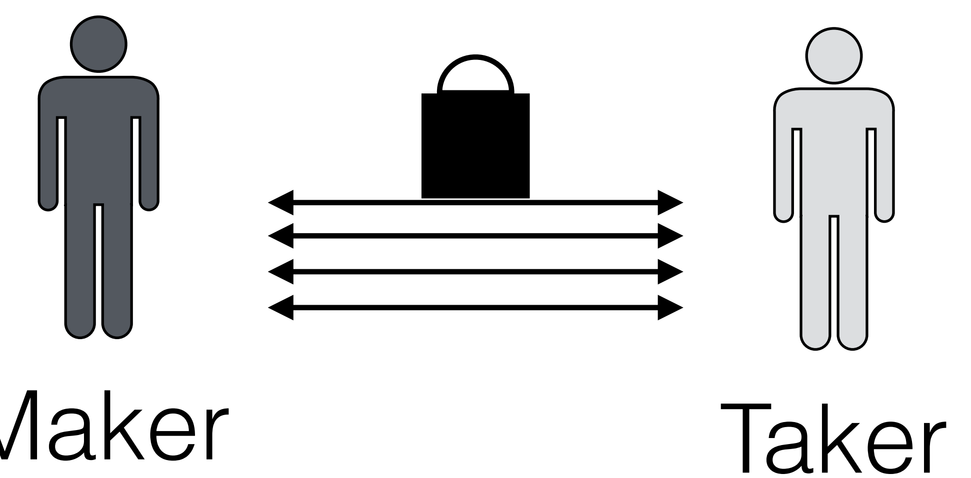
Sell	Buy	Public key
BAT	MKR	pk_1
ZRX	OMG	pk_2
A	B	pk_M



Private Intent-based DEX

Index

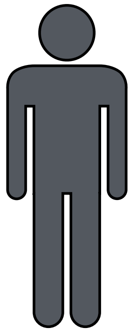
Sell	Buy	Public key
BAT	MKR	pk_1
ZRX	OMG	pk_2
A	B	pk_M



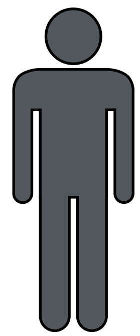
Maker: (A, v_A) (B, v_B)

Taker: (A, v_A) (B, v_B)

Private Intent-based DEX



Private Intent-based DEX



Maker

$\mathbf{c}_1^{\mathsf{old}}$

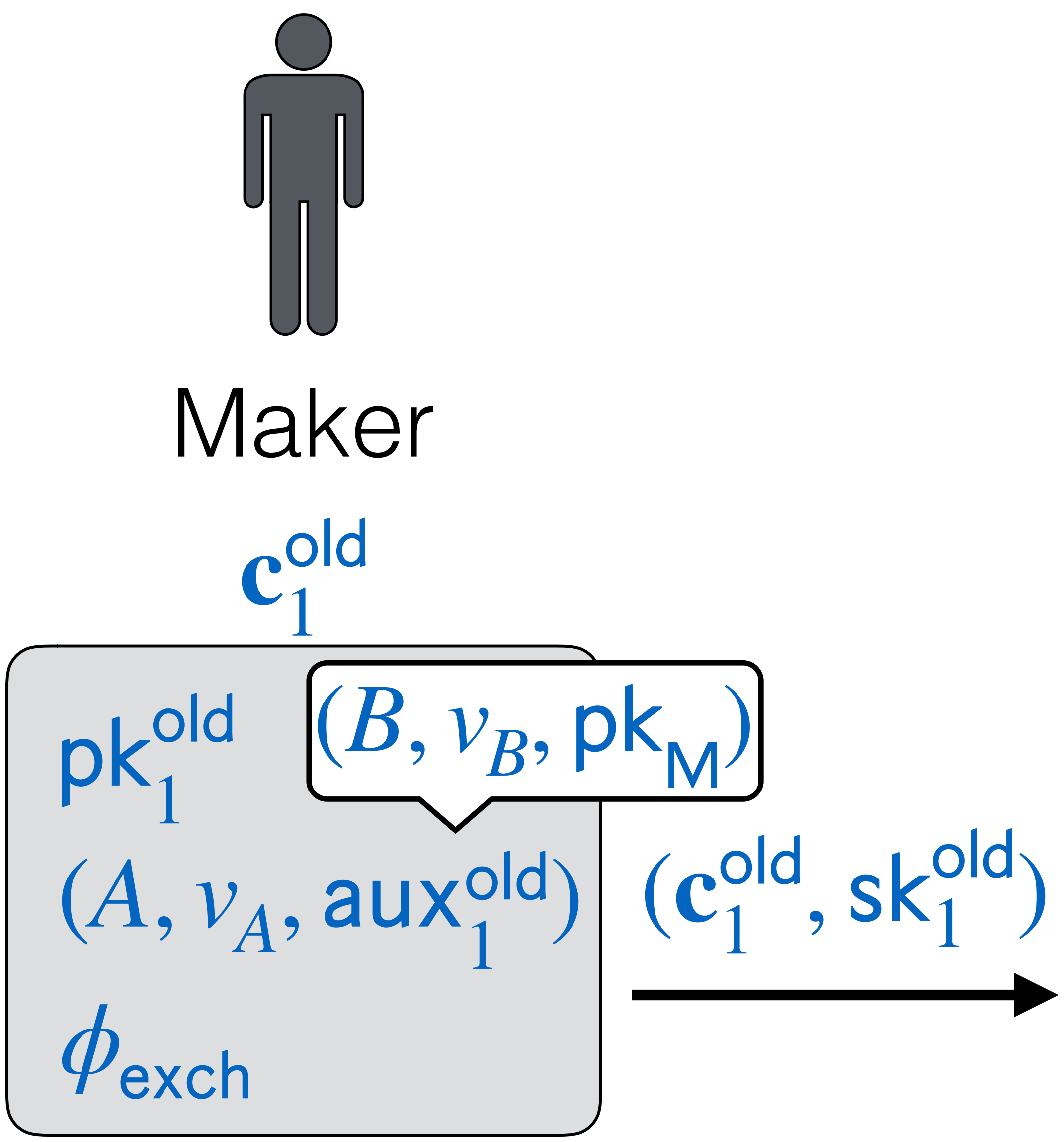
$\mathsf{pk}_1^{\mathsf{old}}$

$(B, v_B, \mathsf{pk}_{\mathsf{M}})$

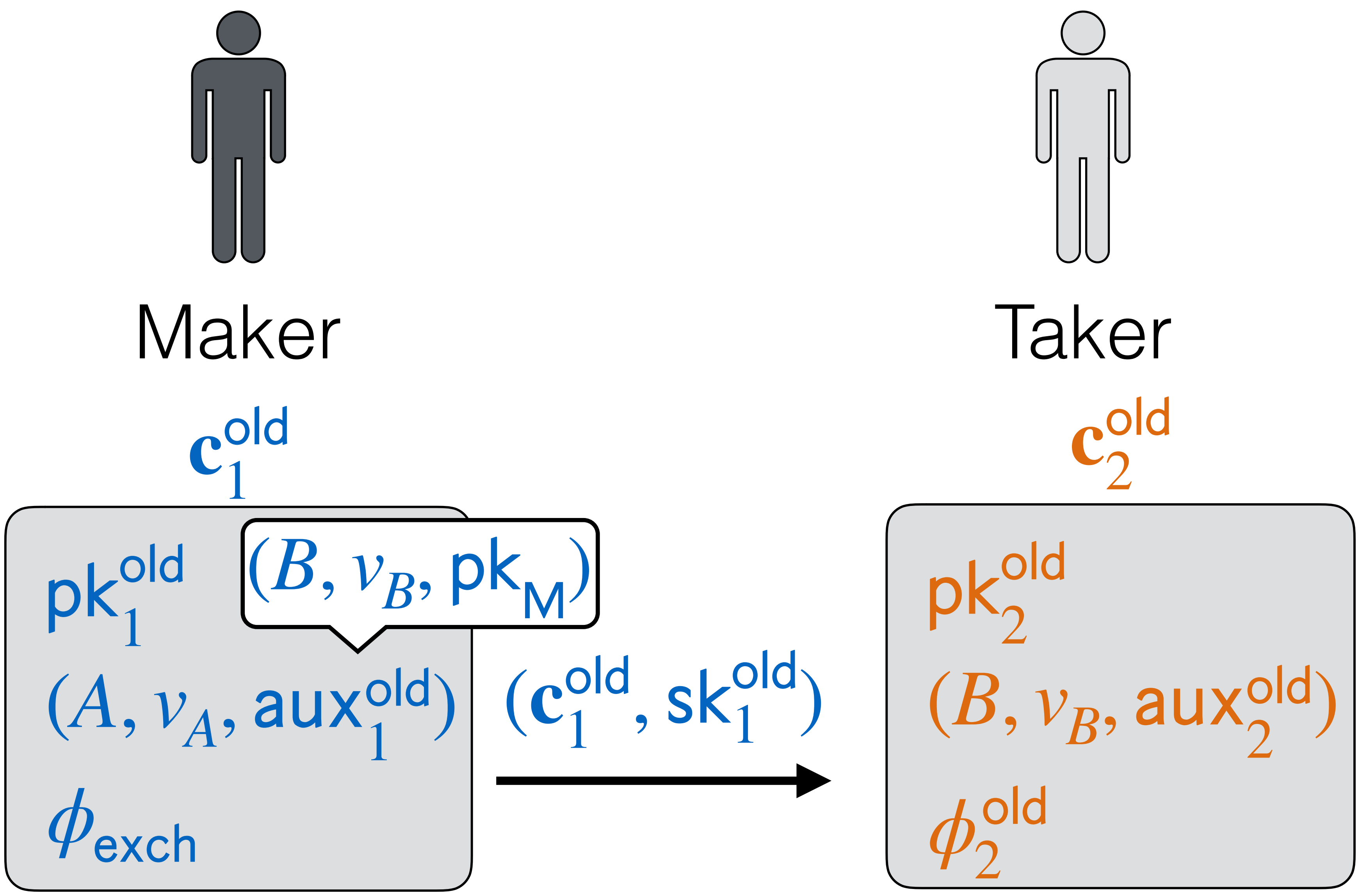
$(A, v_A, \mathsf{aux}_1^{\mathsf{old}})$

ϕ_{exch}

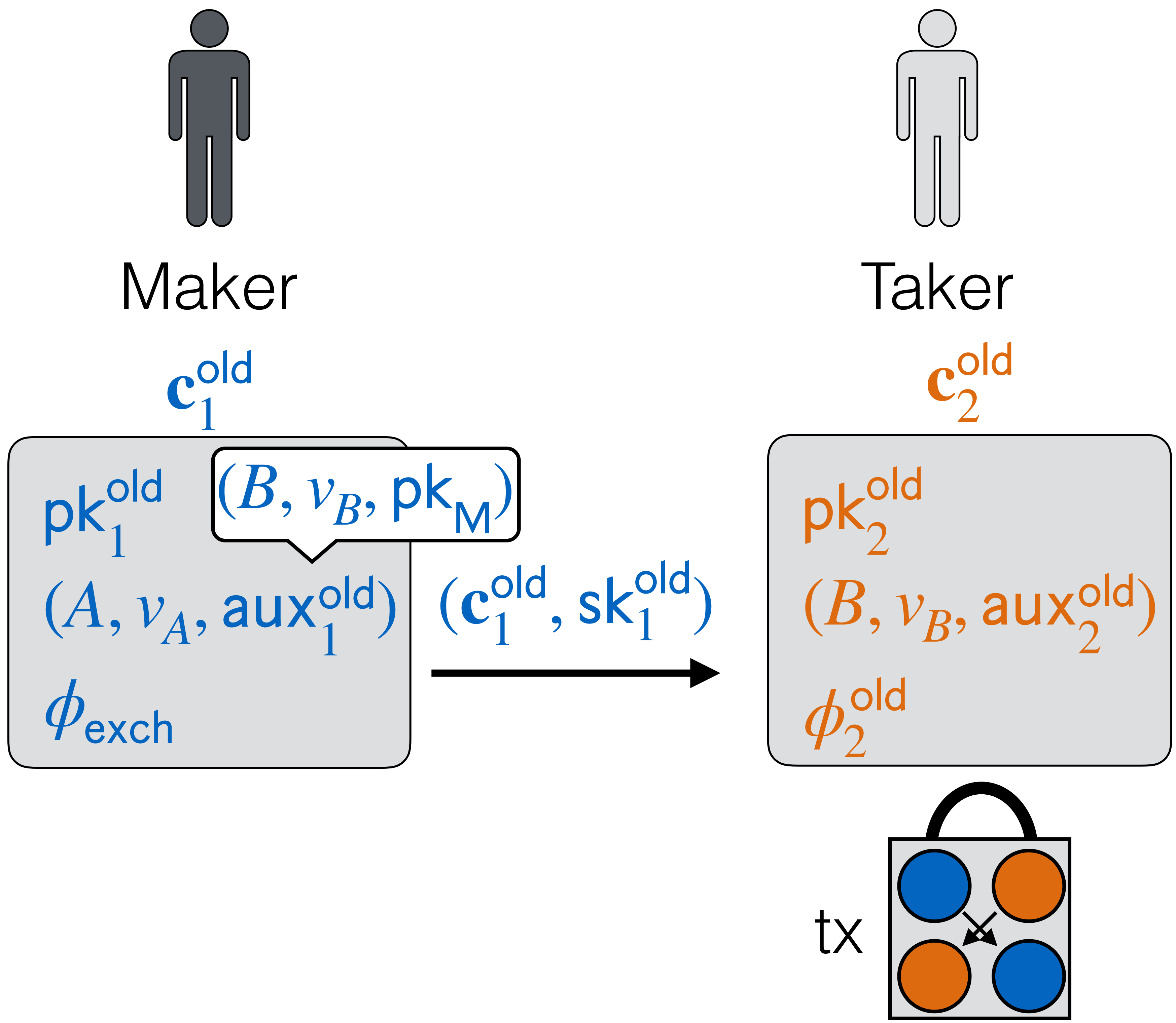
Private Intent-based DEX



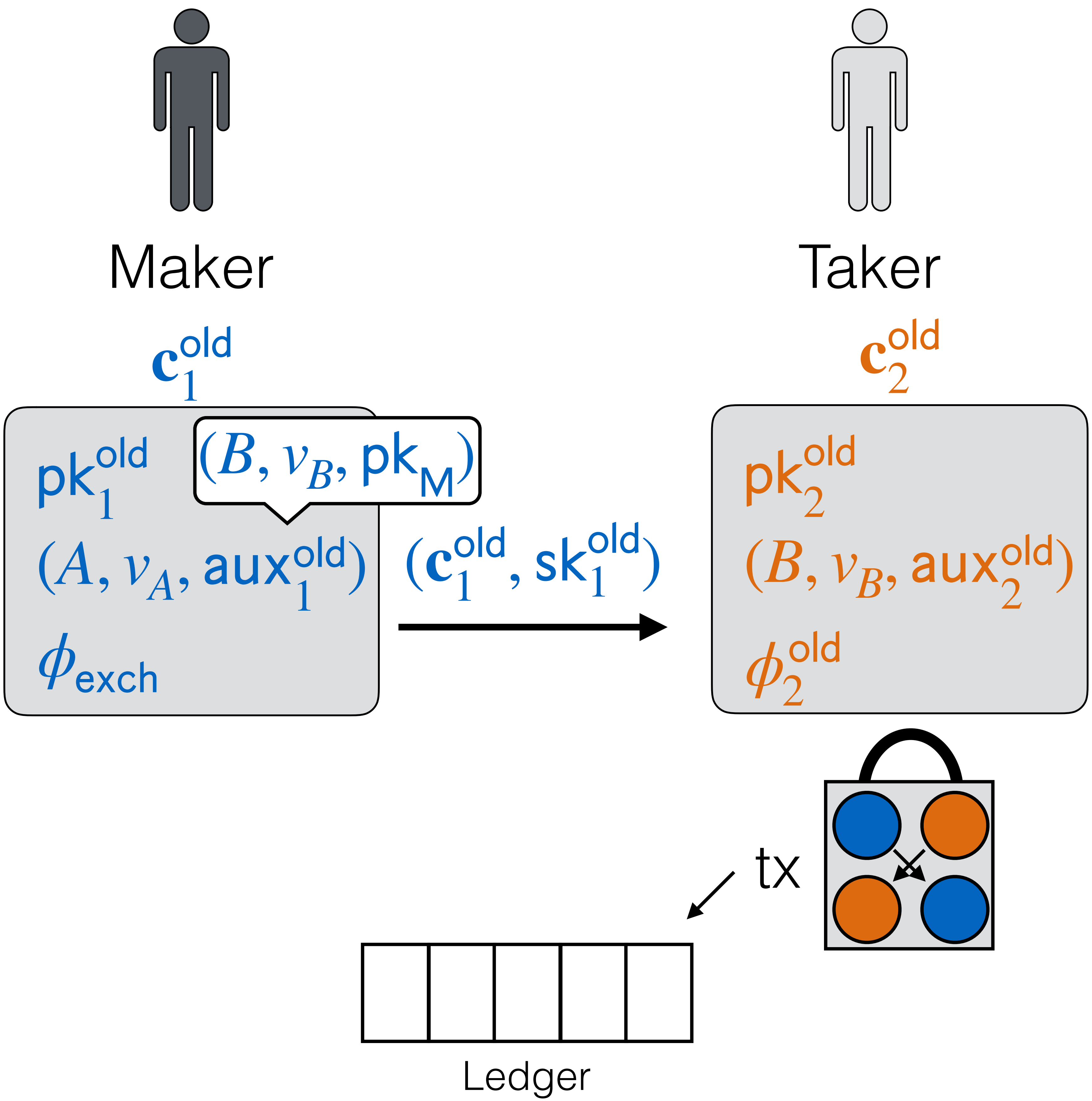
Private Intent-based DEX



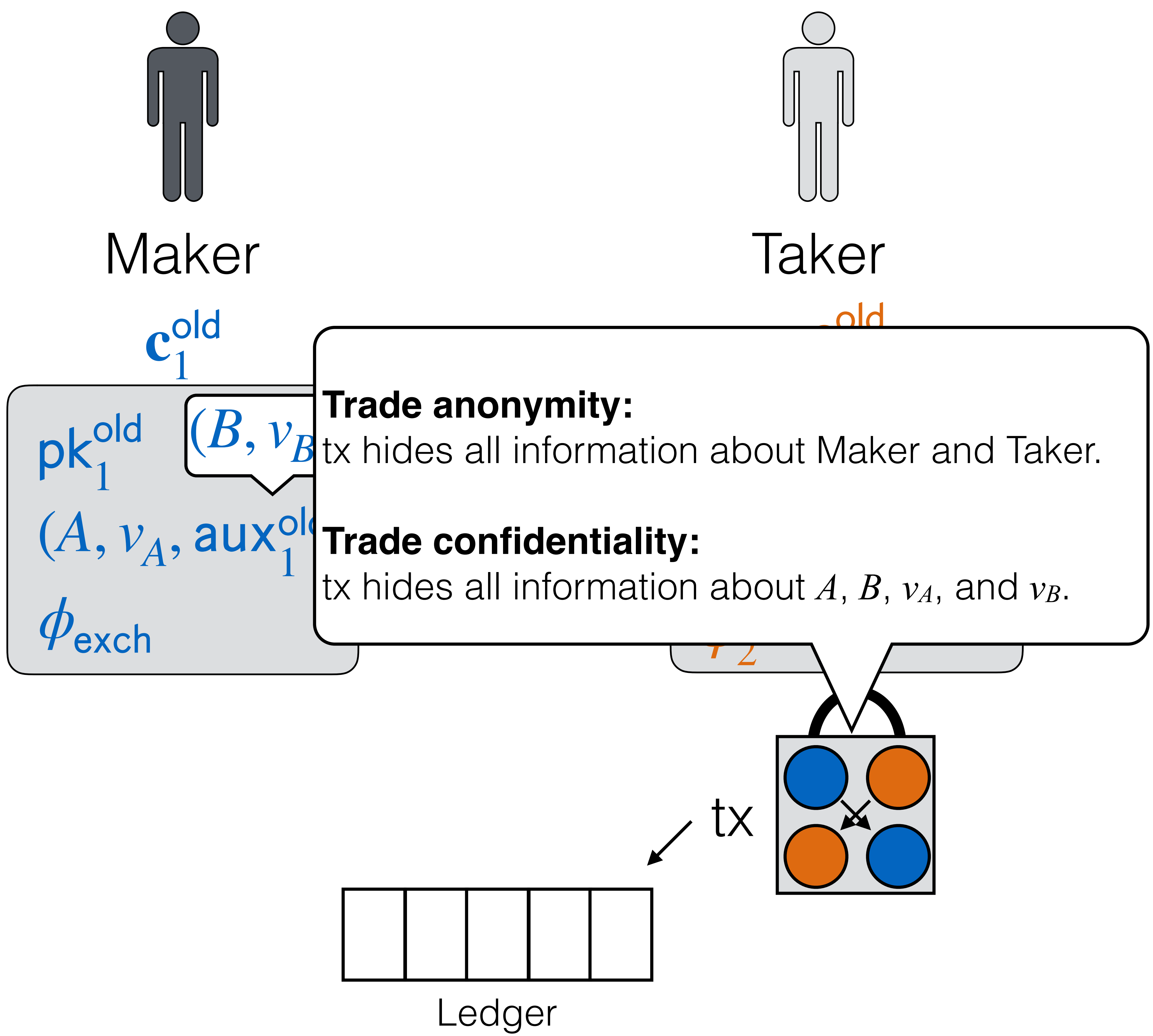
Private Intent-based DEX



Private Intent-based DEX



Private Intent-based DEX



Private user-defined assets

Private user-defined assets

Custom access to private user-defined assets

Private user-defined assets

Custom access to private user-defined assets

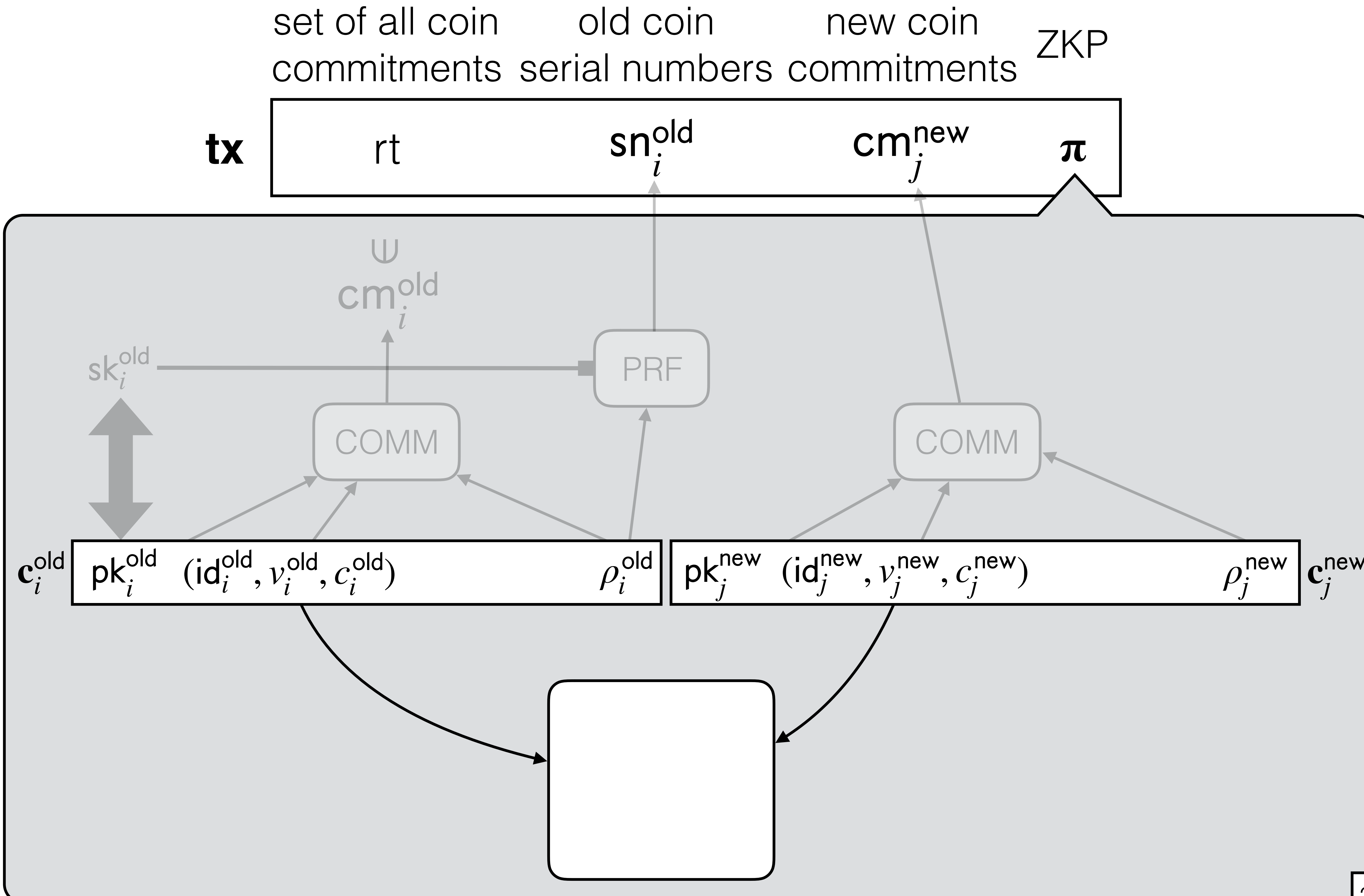
Private DEX

But what if you want a user-defined asset with a different (custom) policy?

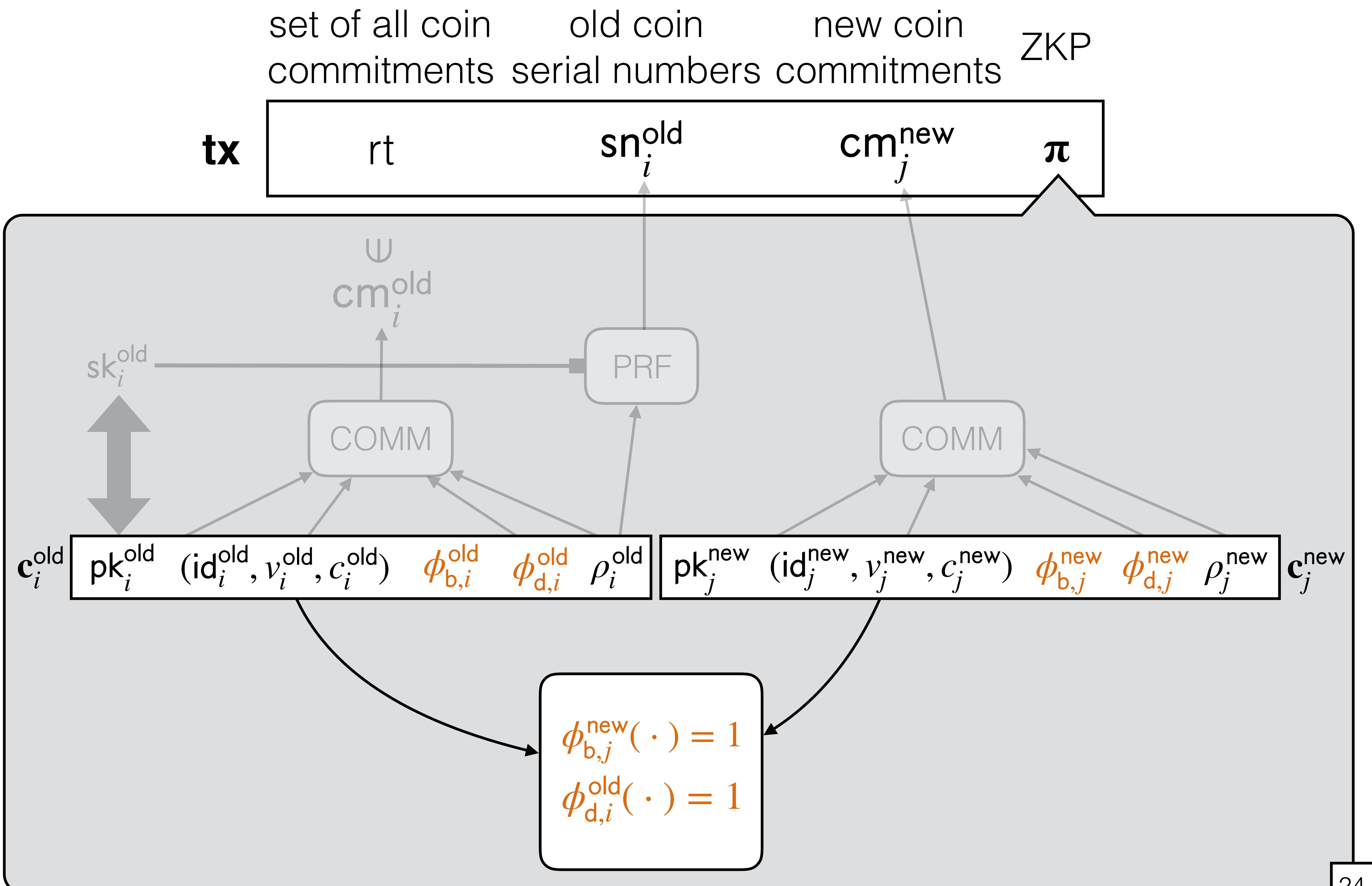
But what if you want a user-defined asset with a different (custom) policy?

For example,
simple ERC-20 tokens require only value-conservation.
other tokens, like stablecoins, need to also implement blacklists and whitelists.

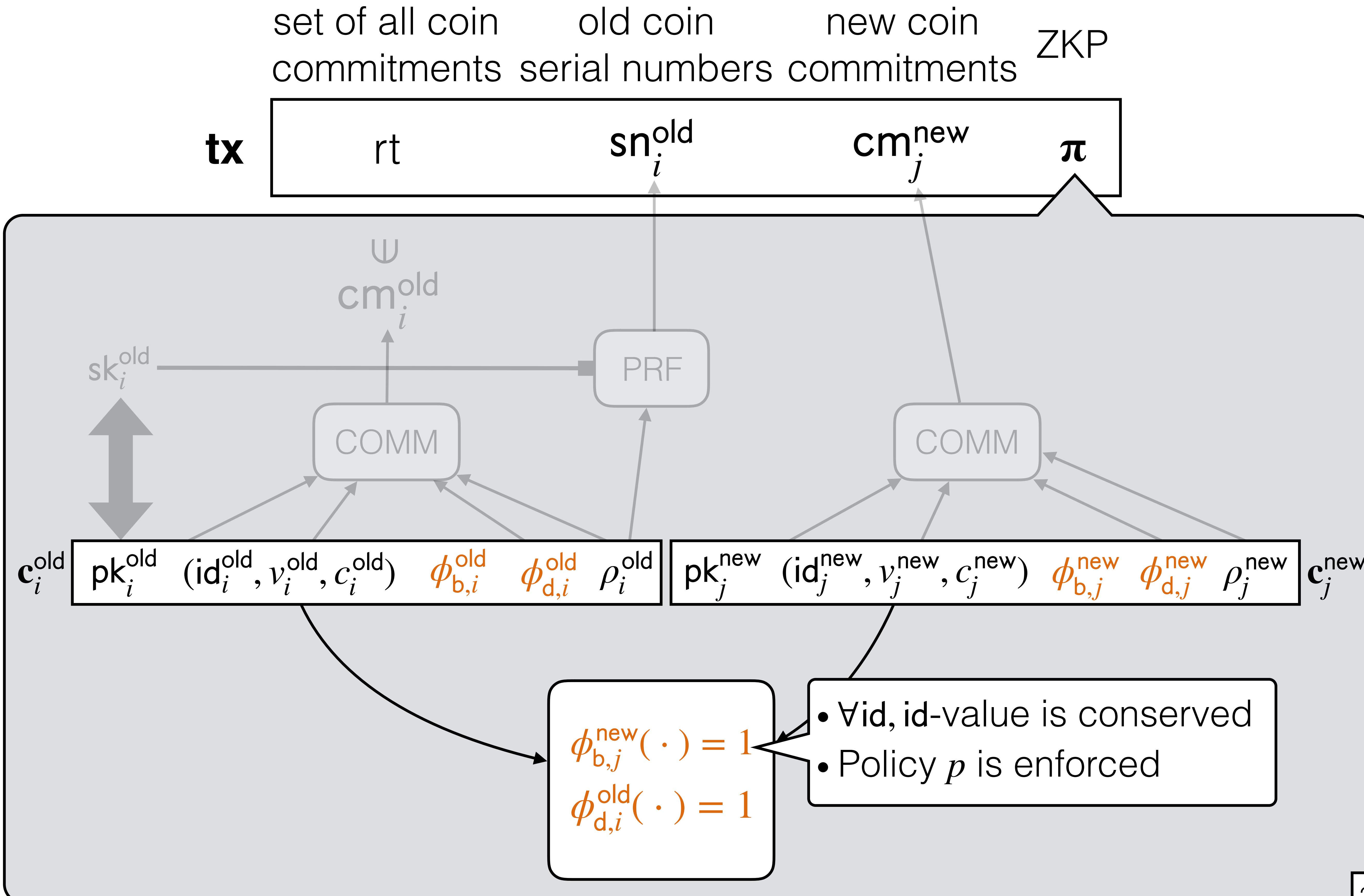
Birth predicates



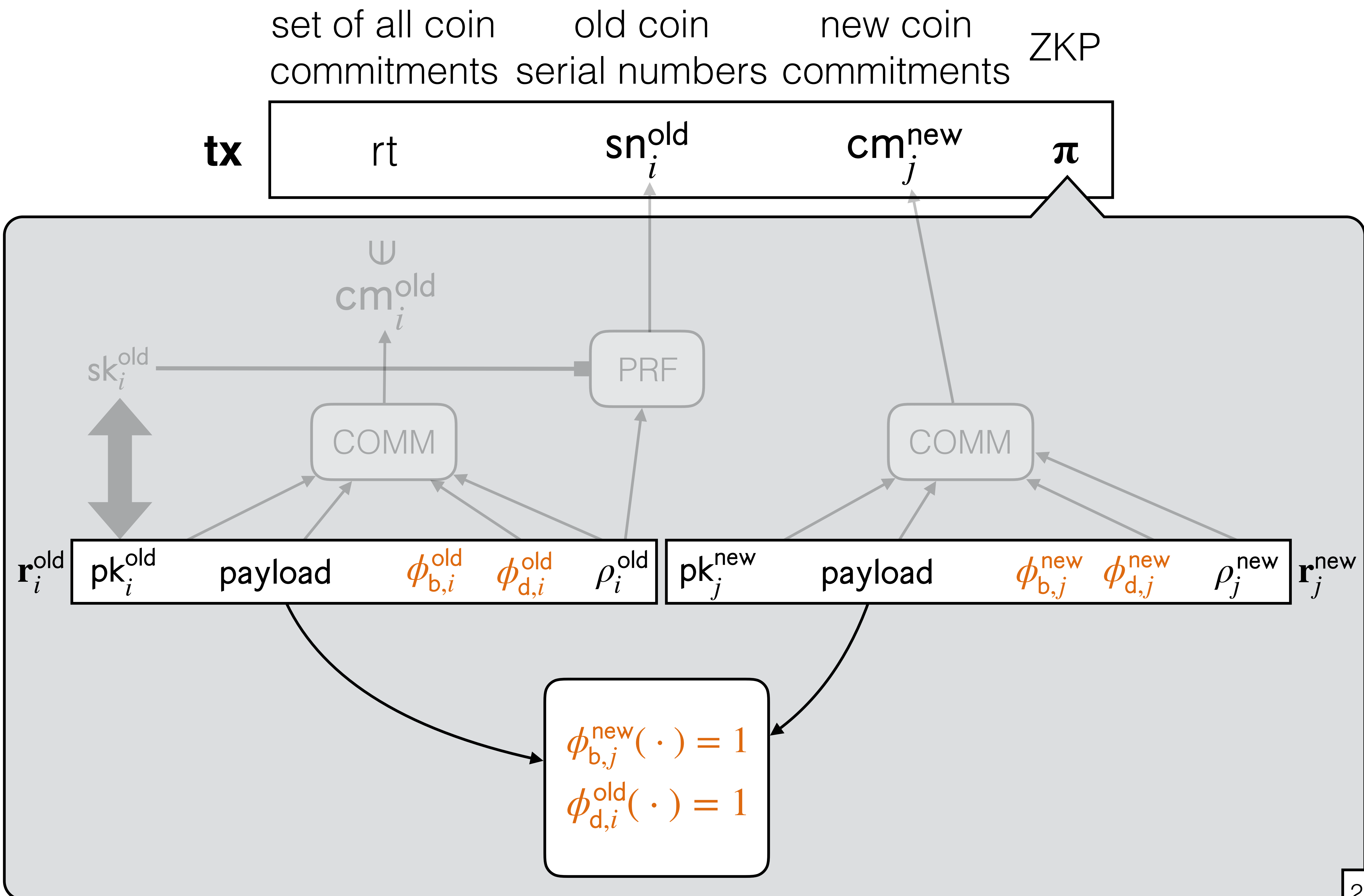
Birth predicates



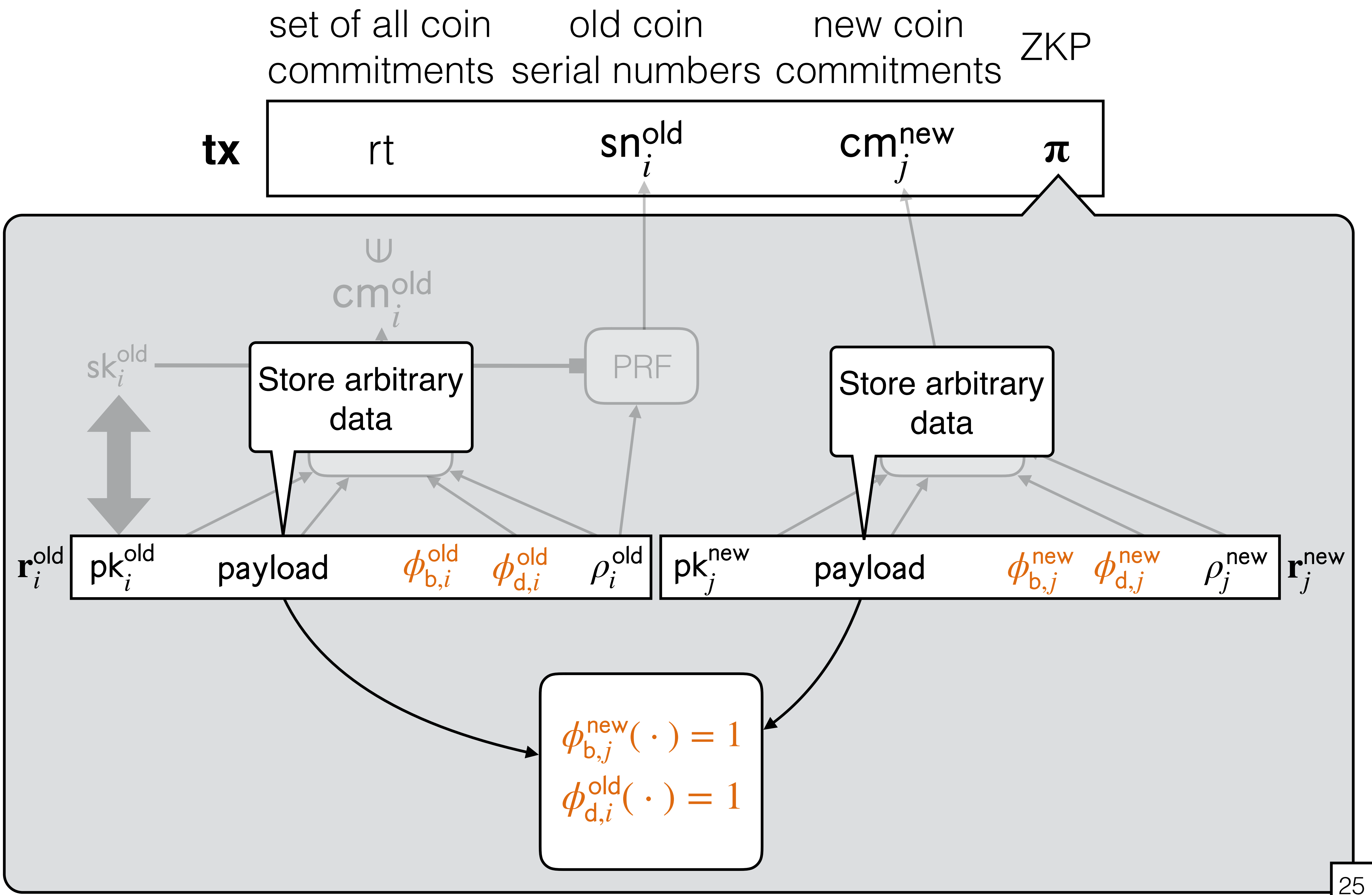
Birth predicates



Supporting arbitrary data and computation



Supporting arbitrary data and computation



ZEXE

ZEXE

Modeling: Records Nano-Kernel

minimalist shared execution environment that defines rules for computing on *records* (units of data)

ZEXE

Modeling: Records Nano-Kernel

minimalist shared execution environment that defines rules for computing on *records* (units of data)

ZEXE

Modeling: Records Nano-Kernel

minimalist shared execution environment that defines rules for computing on *records* (units of data)

Theoretical crypto: Decentralized Private Computation

crypto primitive that realizes a ledger-based RNK where txs reveal NO information about computations

ZEXE

Modeling: Records Nano-Kernel

minimalist shared execution environment that defines rules for computing on *records* (units of data)

↓

Theoretical crypto: Decentralized Private Computation

crypto primitive that realizes a ledger-based RNK where txs reveal NO information about computations

↓

ZEXE

Modeling: Records Nano-Kernel

minimalist shared execution environment that defines rules for computing on *records* (units of data)

↓

Theoretical crypto: Decentralized Private Computation

crypto primitive that realizes a ledger-based RNK where txs reveal NO information about computations

↓

Applied crypto: ZEXE

leverage techniques from zkSNARKs, recursive composition, and efficient circuit design

ZEXE

Modeling: Records Nano-Kernel

minimalist shared execution environment that defines rules for computing on *records* (units of data)

Theoretical crypto: Decentralized Private Computation

crypto primitive that realizes a ledger-based RNK where txs reveal NO information about computations

Applied crypto: ZEXE

leverage techniques from zkSNARKs, recursive composition, and efficient circuit design

libzexe.org

Thanks!

[ia.cr/2018/962](ia.cr/2018/962)

libzexe.org