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

Privacy

- Bottlenecked by weakest node
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:

executed program

input data
Scalability and privacy issues

**Scalability**

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

**Privacy**

Anyone can see:
- executed program
- input data
- caller
Scalability and privacy issues

**Scalability**

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

**Privacy**

Anyone can see:

- Permanently stored on the ledger

- Executed program
- Input data
- Caller
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

\[|tx| < 1\text{KB} \quad \text{Verify}(tx) \text{ in } <50\text{ms}\]
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

\[ |tx| < 1\text{KB} \quad \text{Verify}(tx) \text{ in } <50\text{ms} \]

**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

\[|tx| < 1 \text{KB} \quad \text{Verify}(tx) \text{ in } < 50 \text{ms}\]

**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

\[|tx| < 1\text{KB} \quad \text{Verify}(tx) \text{ in } <50\text{ms}\]

**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

\[|tx| < 1\text{KB} \quad \text{Verify}(tx) \text{ in } <50\text{ms}\]

**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

\[ |tx| < 1\text{KB} \quad \text{Verify}(tx) \text{ in } <50\text{ms} \]

**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

\[ |tx| < 1\text{KB} \quad \text{Verify}(tx) \text{ in } < 50\text{ms} \]

**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: ZEKE
This talk: ZEXE

Will use ideas from ZEXE to construct
This talk: \textcolor{red}{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?

<table>
<thead>
<tr>
<th>Centralized exchanges</th>
<th>Decentralized exchanges</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Users give up custody of assets.</td>
<td></td>
</tr>
<tr>
<td>- Can lead to loss of funds due to fraud/breach/error</td>
<td></td>
</tr>
<tr>
<td>+ Efficient: infrequent on-chain activity</td>
<td></td>
</tr>
<tr>
<td>+ (Somewhat) private: only exchange learns trade details</td>
<td></td>
</tr>
</tbody>
</table>
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
  - 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
  - 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
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 [BCGGMVT14]

Each transaction consumes old coins and creates new coins. The senders, receivers, and values are provably hidden.
The Zerocash Paradigm \cite{BCGGMTV14}

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

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

\[ \text{HIDDEN} \]
The Zerocash Paradigm [BCGGMTV14]

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

\[ V_5 + V_6 = V_e + V_f \]

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

\[ sn_1 \rightarrow cm_a \rightarrow * \rightarrow sn_3 \quad cm_c \quad * \quad sn_5 \rightarrow cm_e \rightarrow * \]

\[ sn_2 \rightarrow cm_b \rightarrow v_5 + v_6 = v_e + v_f \]

all created coins (commitments)
Each transaction consumes old coins and creates new coins. The senders, receivers, and values are provably hidden.

\[ sn_1 \times cm_a \quad sn_3 \times cm_c \quad sn_5 \times cm_e \]

\[ sn_2 \quad cm_b \quad sn_4 \quad cm_d \quad sn_6 \quad cm_f \]

\[ V_5 + V_6 = V_e + V_f \]

**all created coins (commitments)**

\[ cm_b \quad cm_e \quad cm_c \quad ... \]
The Zerocash Paradigm [BCGGMTV14]

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

\[
\begin{align*}
    \text{sn}_1 & \quad \text{cm}_a \\
    \text{sn}_2 & \quad \text{cm}_b \\
    \text{sn}_3 & \quad \text{cm}_c \\
    \text{sn}_4 & \quad \text{cm}_d \\
    \text{sn}_5 & \quad \text{cm}_e \\
    \text{sn}_6 & \quad \text{cm}_f \\
\end{align*}
\]

\[v_5 + v_6 = v_e + v_f\]

all created coins (commitments) \[\text{cm}_b, \text{cm}_c, \ldots\]

consumed coins (serial numbers)
Each transaction consumes old coins and creates new coins. The senders, receivers, and values are provably hidden.

\[ \text{all created coins (commitments)} \quad \text{consumed coins (serial numbers)} \]

\[ sn_1 \text{ cm}_a \quad \ldots \quad sn_2 \text{ cm}_b, \ldots \]
\[ sn_3 \text{ cm}_c \quad \ldots \quad sn_4 \text{ cm}_d \]
\[ sn_5 \text{ cm}_e \quad \ldots \quad sn_6 \text{ cm}_f \]

\[ \text{HIDDEN} \quad V_5 + V_6 = V_e + V_f \]
The Zerocash Paradigm [BCGGMTV14]

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

\[
\begin{align*}
&\text{sn}_1 \quad \text{cm}_a \\
&\text{sn}_2 \quad \text{cm}_b \\
&\text{sn}_3 \quad \text{cm}_c \\
&\text{sn}_4 \quad \text{cm}_d \\
\end{align*}
\]

\[
\begin{align*}
&\text{sn}_5 \quad \text{cm}_e \\
&\text{sn}_6 \quad \text{cm}_f \\
\end{align*}
\]

\[v_5 + v_6 = v_e + v_f\]

**all created** coins (commitments)

\[
\begin{align*}
&\text{cm}_b \\
&\text{cm}_e \\
&\text{cm}_c \\
&\ldots \\
\end{align*}
\]

**consumed** coins (serial numbers)

\[
\begin{align*}
&\text{sn}_5 \\
&\ldots \\
\end{align*}
\]
Sketch of Zerocash

old coin  new coin
serial numbers commitments

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

<table>
<thead>
<tr>
<th>tx</th>
<th>$\text{sn}_i^{\text{old}}$</th>
<th>$\text{cm}_j^{\text{new}}$</th>
<th>$\pi$</th>
</tr>
</thead>
</table>

- old coin serial numbers
- new coin commitments
- ZKP

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.

[BCGGMTV14]
Sketch of Zerocash

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

<table>
<thead>
<tr>
<th>tx</th>
<th>rt</th>
<th>( \pi )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \psi )</td>
<td>( \psi )</td>
<td>( \psi )</td>
</tr>
</tbody>
</table>

| set of all coin commitments | old coin serial numbers | new coin commitments |

ZKP

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

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

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

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.
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

- set of all coin commitments
- old coin serial numbers
- new coin commitments
- ZKP

\[ \text{tx} \]
\[ \begin{align*}
rt & \quad \text{sn}_{i}^{\text{old}} & \quad \text{cm}_{j}^{\text{new}} & \quad \pi
\end{align*} \]

\[ \Psi \]
\[ \text{cm}_{i}^{\text{old}} \]
\[ \text{sk}_{i}^{\text{old}} \]
\[ \text{pk}_{i}^{\text{old}} \]
\[ \rho_{i}^{\text{old}} \]
\[ \text{Comm} \]

\[ \text{pk}_{j}^{\text{new}} \]
\[ \rho_{j}^{\text{new}} \]
\[ \text{cm}_{j}^{\text{new}} \]

* For now we ignore how to mint an initial supply and generate a unique id.
Private user-defined assets

set of all coin commitments
old coin serial numbers
new coin commitments

\[ \text{tx} \quad \begin{array}{cccc}
\text{rt} & \text{sn}^\text{old}_i & \text{cm}^\text{new}_j & \pi \\
\end{array} \]

\[ \Psi \quad \text{cm}^\text{old}_i \quad \text{PRF} \quad \text{COMM} \quad \text{cm}^\text{old}_i \quad \text{PRF} \quad \text{COMM} \]

\[ \text{pk}^\text{old}_i \quad (\text{id}^\text{old}_i, \nu^\text{old}_i) \quad \rho^\text{old}_i \quad \text{pk}^\text{new}_j \quad (\text{id}^\text{new}_j, \nu^\text{new}_j) \quad \rho^\text{new}_j \]

* 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

- Set of all coin commitments
- Original coin serial numbers
- New coin commitments

**tx**
- **rt**
- **sn\textsuperscript{old}**
- **cm\textsuperscript{new}**
- **\pi**

- **sk\textsuperscript{old}_i**
- **pk\textsuperscript{old}_i (id\textsuperscript{old}_i, v\textsuperscript{old}_i, aux\textsuperscript{old}_i)**
- **\rho\textsuperscript{old}_i**

- **Ψ**
- **cm\textsuperscript{old}_i**

- **PRF**

- **pk\textsuperscript{new}_j (id\textsuperscript{new}_j, v\textsuperscript{new}_j, aux\textsuperscript{new}_j)**
- **\rho\textsuperscript{new}_j**

- **cm\textsuperscript{new}_j**

- **\forall id, id-value is conserved**
Custom access to user-defined assets

- Set of all coin commitments
- Old coin serial numbers
- New coin commitments
- ZKP

\[ \text{tx} \quad \text{rt} \quad \text{sn}_{old}^{i} \quad \text{cm}_{new}^{j} \quad \pi \]

\[ \psi \quad \text{cm}_{old}^{i} \]

\[ \text{sk}_{i}^{old} \]

\[ \text{pk}_{i}^{old} (id_{i}^{old}, v_{i}^{old}, aux_{i}^{old}) \quad \phi_{i}^{old} \quad \rho_{i}^{old} \]

\[ \text{pk}_{j}^{new} (id_{j}^{new}, v_{j}^{new}, aux_{j}^{new}) \quad \phi_{j}^{new} \quad \rho_{j}^{new} \]

- \( \forall id, \text{id-value is conserved} \)
- \( \forall i, \phi_{i}^{old}(\text{all coins}) = 1 \)
Custom access to user-defined assets

- Set of all coin commitments
- Old coin serial numbers
- New coin commitments
- ZKP

Transaction reveals no information about asset identifier, value, or predicate

- ∀i, id-value is conserved
- ∀i, $\phi_i^{\text{old}}(\text{all coins}) = 1$
Private atomic swaps
Private atomic swaps

Key primitive for constructing DEXs
Private atomic swaps

Key primitive for constructing DEXs

\[ \phi_1^{\text{old}} = \phi_{\text{exch}} \]
Private atomic swaps

Key primitive for constructing DEXs
Private atomic swaps

Key primitive for constructing DEXs

\[ \phi_{\text{exch}}(c_1^{\text{old}}, c_2^{\text{old}}, c_1^{\text{new}}, c_2^{\text{new}}) : \]
Private atomic swaps

Key primitive for constructing DEXs

\[ \phi_{\text{exch}}(c_1^\text{old}, c_2^\text{old}, c_1^\text{new}, c_2^\text{new}) : \]

Parse \( \text{aux}_1^\text{old} \) as \( (id^*, \nu^*, pk^*) \)
Private atomic swaps

Key primitive for constructing DEXs

\[
\phi_{\text{exch}}(c_1^{\text{old}}, c_2^{\text{old}}, c_1^{\text{new}}, c_2^{\text{new}}):
\]

Parse \( aux_1^{\text{old}} \) as \((id^*, \nu^*, pk^*)\)

Check that \( id_2^{\text{old}} = id^* \) and \( \nu_2^{\text{old}} = \nu^* \)
Private atomic swaps

Key primitive for constructing DEXs

\[ \phi_{\text{exch}}(c_1^{\text{old}}, c_2^{\text{old}}, c_1^{\text{new}}, c_2^{\text{new}}) : \]

Parse \( \text{aux}_1^{\text{old}} \) as \((\text{id}^*, \nu^*, \text{pk}^*)\)

Check that \( \text{id}_2^{\text{old}} = \text{id}^* \) and \( \nu_2^{\text{old}} = \nu^* \)

Perform swap:

\[ \begin{align*}
\text{id}_1^{\text{new}} &= \text{id}_2^{\text{old}}; & \nu_1^{\text{new}} &= \nu_2^{\text{old}} \\
\text{id}_2^{\text{new}} &= \text{id}_1^{\text{old}}; & \nu_2^{\text{new}} &= \nu_1^{\text{old}}
\end{align*} \]
Private atomic swaps

Key primitive for constructing DEXs

\[
\phi_{\text{exch}}(c_1^{\text{old}}, c_2^{\text{old}}, c_1^{\text{new}}, c_2^{\text{new}}) :
\]
Parse \(\text{aux}_1^{\text{old}}\) as \((\text{id}^*, \nu^*, \text{pk}^*)\)
Check that \(\text{id}_2^{\text{old}} = \text{id}^*\) and \(\nu_2^{\text{old}} = \nu^*\)
Perform swap:
\[
\begin{align*}
\text{id}_1^{\text{new}} &= \text{id}_2^{\text{old}}; & \nu_1^{\text{new}} &= \nu_2^{\text{old}} \\
\text{id}_2^{\text{new}} &= \text{id}_1^{\text{old}}; & \nu_2^{\text{new}} &= \nu_1^{\text{old}}
\end{align*}
\]
Check addresses:
\[
\begin{align*}
\text{pk}_1^{\text{new}} &= \text{pk}^* \\
\text{pk}_2^{\text{new}} &= \text{pk}_2^{\text{old}}
\end{align*}
\]
DEX
DEX \rightarrow \text{Atomic swap} \checkmark
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: $\langle A, B, v_A, v_B, \ldots \rangle$

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

Order-based: \((A, B, v_A, v_B, ...)\)

- *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: $(A, B, v_A, v_B, ...)$

- *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: 

• *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: 

- *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
- Takers can scan for intentions and interact with makers to fill orders
Order-based: 

• *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
• Takers can scan for intentions and interact with makers to fill orders
• Eg: AirSwap
**DEX Architectures**

**Order-based:**

- *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
- Takers can scan for intentions and interact with makers to fill orders
- Eg: AirSwap
## Intent-based DEX

### Index

<table>
<thead>
<tr>
<th>Sell</th>
<th>Buy</th>
<th>Public key</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAT</td>
<td>MKR</td>
<td>pk₁</td>
</tr>
<tr>
<td>ZRX</td>
<td>OMG</td>
<td>pk₂</td>
</tr>
</tbody>
</table>
## Intent-based DEX

<table>
<thead>
<tr>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sell</td>
</tr>
<tr>
<td>BAT</td>
</tr>
<tr>
<td>ZRX</td>
</tr>
</tbody>
</table>

Maker
## Intent-based DEX

### Index

<table>
<thead>
<tr>
<th>Sell</th>
<th>Buy</th>
<th>Public key</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAT</td>
<td>MKR</td>
<td>pk_1</td>
</tr>
<tr>
<td>ZRX</td>
<td>OMG</td>
<td>pk_2</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>pk_M</td>
</tr>
</tbody>
</table>

**Maker**
## Intent-based DEX

### Index

<table>
<thead>
<tr>
<th>Sell</th>
<th>Buy</th>
<th>Public key</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAT</td>
<td>MKR</td>
<td>pk₁</td>
</tr>
<tr>
<td>ZRX</td>
<td>OMG</td>
<td>pk₂</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>pk₉</td>
</tr>
</tbody>
</table>

**Maker**

**Taker**
### Intent-based DEX

<table>
<thead>
<tr>
<th>Sell</th>
<th>Buy</th>
<th>Public key</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAT</td>
<td>MKR</td>
<td>pk_1</td>
</tr>
<tr>
<td>ZRX</td>
<td>OMG</td>
<td>pk_2</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>pk_M</td>
</tr>
</tbody>
</table>

**Index**

**Maker**

**Taker**
## Intent-based DEX

### Index

<table>
<thead>
<tr>
<th>Sell</th>
<th>Buy</th>
<th>Public key</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAT</td>
<td>MKR</td>
<td>pk₁</td>
</tr>
<tr>
<td>ZRX</td>
<td>OMG</td>
<td>pk₂</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>pkₐ</td>
</tr>
</tbody>
</table>

**Diagram:**
- Maker
- Taker
- tx

**Notes:**
- BAT and MKR
- ZRX and OMG
- A and B
- pk₁, pk₂, pkₐ
## Intent-based DEX

### Index

<table>
<thead>
<tr>
<th>Sell</th>
<th>Buy</th>
<th>Public key</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAT</td>
<td>MKR</td>
<td>pk(_1)</td>
</tr>
<tr>
<td>ZRX</td>
<td>OMG</td>
<td>pk(_2)</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>pk(_M)</td>
</tr>
</tbody>
</table>

- **Maker**: Arrow with bag pointing to ledger
- **Taker**: Arrow with.sketch pointing to ledger
- **tx**: Transaction sketch
Privacy Leakage

Ledger

M  T

tx
Privacy Leakage

“The addresses \( \text{addr}_M \) and \( \text{addr}_T \) are exchanging \( v_A \) units of A for \( v_B \) units of B”
Privacy Leakage

The addresses \texttt{addr}_M and \texttt{addr}_T are exchanging \(v_A\) units of A for \(v_B\) units of B.”
Privacy Leakage

“The addresses \( \text{addr}_M \) and \( \text{addr}_T \) are exchanging \( v_A \) units of A for \( v_B \) units of B.”

 identities of transacting parties

assets and values in the trade

Ledger
The addresses \( \text{addr}_M \) and \( \text{addr}_T \) are exchanging \( v_A \) units of A for \( v_B \) units of B. 

**Trade anonymity**

No information about participants in the trade is revealed.
Privacy Leakage

“The addresses $\text{addr}_M$ and $\text{addr}_T$ are exchanging $v_A$ units of A for $v_B$ 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

<table>
<thead>
<tr>
<th>Sell</th>
<th>Buy</th>
<th>Public key</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAT</td>
<td>MKR</td>
<td>pk₁</td>
</tr>
<tr>
<td>ZRX</td>
<td>OMG</td>
<td>pk₂</td>
</tr>
</tbody>
</table>
## Private Intent-based DEX

### Index

<table>
<thead>
<tr>
<th>Sell</th>
<th>Buy</th>
<th>Public key</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAT</td>
<td>MKR</td>
<td>pk₁</td>
</tr>
<tr>
<td>ZRX</td>
<td>OMG</td>
<td>pk₂</td>
</tr>
</tbody>
</table>

**Maker**
## Private Intent-based DEX

### Index

<table>
<thead>
<tr>
<th>Sell</th>
<th>Buy</th>
<th>Public key</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAT</td>
<td>MKR</td>
<td>pk(_1)</td>
</tr>
<tr>
<td>ZRX</td>
<td>OMG</td>
<td>pk(_2)</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>pk(_M)</td>
</tr>
</tbody>
</table>
# Private Intent-based DEX

## Index

<table>
<thead>
<tr>
<th>Sell</th>
<th>Buy</th>
<th>Public key</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAT</td>
<td>MKR</td>
<td>pk₁</td>
</tr>
<tr>
<td>ZRX</td>
<td>OMG</td>
<td>pk₂</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>pkₘ</td>
</tr>
</tbody>
</table>

### Maker

![Maker](image)

### Taker

![Taker](image)
## Private Intent-based DEX

### Index

<table>
<thead>
<tr>
<th>Sell</th>
<th>Buy</th>
<th>Public key</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAT</td>
<td>MKR</td>
<td>pk₁</td>
</tr>
<tr>
<td>ZRX</td>
<td>OMG</td>
<td>pk₂</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>pkₘ</td>
</tr>
</tbody>
</table>

**Diagram:**

- **Maker**
- **Taker**
Private Intent-based DEX

Index

<table>
<thead>
<tr>
<th>Sell</th>
<th>Buy</th>
<th>Public key</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAT</td>
<td>MKR</td>
<td>pk₁</td>
</tr>
<tr>
<td>ZRX</td>
<td>OMG</td>
<td>pk₂</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>pkₘ</td>
</tr>
</tbody>
</table>

\[ (A, v_A) \quad (B, v_B) \]

Maker

\[(A, v_A) \quad (B, v_B)\]

Taker
Private Intent-based DEX

Maker
Private Intent-based DEX

Maker

$c_1^{\text{old}}$

$(A, \nu_A, \text{aux}_1^{\text{old}})$

$\phi_{\text{exch}}$

$pk_1^{\text{old}}$

$(B, \nu_B, pk_M)$
Private Intent-based DEX

Maker

\[ c_1^{\text{old}} \]
\[ (B, \nu_B, \text{pk}_M) \]
\[ (A, \nu_A, \text{aux}_1^{\text{old}}) \]
\[ \phi_{\text{exch}} \]

Taker

\[ (c_1^{\text{old}}, \text{sk}_1^{\text{old}}) \]
Private Intent-based DEX

**Maker**

\[\phi_{\text{exch}} \]

\[\text{pk}^\text{old}_1 \]

\[(A, v_A, \text{aux}^\text{old}_1)\]

\[c^\text{old}_1\]

**Taker**

\[\phi^\text{old}_2\]

\[\text{pk}^\text{old}_2\]

\[(B, v_B, \text{aux}^\text{old}_2)\]

\[c^\text{old}_2\]

\[(\text{B, v}_B, \text{pk}_M)\]
Private Intent-based DEX

Maker

\[ c_1^{\text{old}} \]

\[ (A, \nu_A, \text{aux}_1^{\text{old}}) \]

\[ \phi_{\text{exch}} \]

\[ \text{pk}_1^{\text{old}} \]

\[ (B, \nu_B, \text{pk}_M) \]

Taker

\[ c_2^{\text{old}} \]

\[ (B, \nu_B, \text{aux}_2^{\text{old}}) \]

\[ \text{pk}_2^{\text{old}} \]

\[ \phi_2^{\text{old}} \]

\[ \text{tx} \]
Private Intent-based DEX

**Maker**

- $c_1^{\text{old}}$
- $pk_1^{\text{old}}$
- $(A, v_A, aux_1^{\text{old}})$
- $\phi_{\text{exch}}$

**Taker**

- $c_2^{\text{old}}$
- $pk_2^{\text{old}}$
- $(B, v_B, aux_2^{\text{old}})$
- $\phi_2^{\text{old}}$

Transaction: $tx$

Ledger
Private Intent-based DEX

Trade anonymity:
- tx hides all information about Maker and Taker.

Trade confidentiality:
- tx hides all information about $A$, $B$, $v_A$, and $v_B$. 
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

set of all coin commitments
old coin serial numbers
new coin commitments

\[ \text{tx} \]

\[ \text{rt} \]

\[ \text{sn}^{\text{old}}_i \]

\[ \text{cm}^{\text{new}}_j \]

\[ \pi \]

\[ \Psi \]

\[ \text{cm}^{\text{old}}_i \]

\[ \text{sk}^{\text{old}}_i \]

\[ \text{pk}^{\text{old}}_i (\text{id}^{\text{old}}_i, v^{\text{old}}_i, c^{\text{old}}_i) \]

\[ \rho_i^{\text{old}} \]

\[ \text{pk}^{\text{new}}_j (\text{id}^{\text{new}}_j, v^{\text{new}}_j, c^{\text{new}}_j) \]

\[ \rho_j^{\text{new}} \]

\[ c^{\text{new}}_j \]
Birth predicates

- Set of all coin commitments
- Old coin serial numbers
- New coin commitments
- ZKP

\(\text{tx} \quad \text{rt} \quad \text{sn}^{\text{old}}_i \quad \text{cm}^{\text{new}}_j \quad \pi\)

\[\text{pk}^{\text{old}}_i (\text{id}^{\text{old}}_i, v^{\text{old}}_i, c^{\text{old}}_i) \quad \phi^{\text{old}}_{b,i} \quad \phi^{\text{old}}_{d,i} \quad \rho^{\text{old}}_i \quad \text{comm}_i \]

\[\text{pk}^{\text{new}}_j (\text{id}^{\text{new}}_j, v^{\text{new}}_j, c^{\text{new}}_j) \quad \phi^{\text{new}}_{b,j} \quad \phi^{\text{new}}_{d,j} \quad \rho^{\text{new}}_j \quad \text{comm}_j\]

\[\psi \quad \text{cm}^{\text{old}}_i \quad \text{sk}^{\text{old}}_i\]

\[\phi^{\text{new}}_{b,j} (\cdot) = 1 \quad \phi^{\text{old}}_{d,i} (\cdot) = 1\]
Birth predicates

<table>
<thead>
<tr>
<th>tx</th>
<th>rt</th>
<th>sn\textsuperscript{old}</th>
<th>cm\textsuperscript{new}</th>
<th>( \pi )</th>
</tr>
</thead>
</table>

- set of all coin commitments
- old coin serial numbers commitments
- new coin commitments

\( \text{ZKP} \)

\( \text{PRF} \)

\( \text{COMM} \)

\( \text{COMM} \)

\( \psi \)

\( \text{PRF} \)

\( \text{COMM} \)

\( \text{COMM} \)

\( \text{PK}_{\text{old}} \)

\( \text{ID}_{\text{old}}, \text{v}_{\text{old}}, \text{c}_{\text{old}} \)

\( \phi_{\text{b},i}, \phi_{\text{d},i}, \rho_{i} \)

\( \text{PK}_{\text{new}} \)

\( \text{ID}_{\text{new}}, \text{v}_{\text{new}}, \text{c}_{\text{new}} \)

\( \phi_{\text{b},j}, \phi_{\text{d},j}, \rho_{j} \)

\( \text{sk}_{\text{i}} \)

\( \text{sk}_{\text{j}} \)

\( \text{CM}_{\text{i}} \)

\( \text{CM}_{\text{j}} \)

\( \psi \)

\( \text{CM}_{\text{i}} \)

\( \text{CM}_{\text{j}} \)

\( \forall \text{id}, \text{id-value is conserved} \)

\( \text{Policy } p \text{ is enforced} \)

\( \phi_{\text{b},j}(\cdot) = 1 \)

\( \phi_{\text{d},i}(\cdot) = 1 \)
Supporting arbitrary data and computation

(tx, rt, sn_{old}, cm_{new}, \pi)

set of all coin commitments
old coin serial numbers
new coin commitments
ZKP

\psi

\sum cm_{old}^{i}

\sum pk_{i}^{old} \rightarrow \phi_{b,i}^{old} \phi_{d,i}^{old} \rho_{i}^{old}

\sum pk_{j}^{new} \rightarrow \phi_{b,j}^{new} \phi_{d,j}^{new} \rho_{j}^{new}

\phi_{b,j}^{new}(\cdot) = 1
\phi_{d,i}^{old}(\cdot) = 1
Supporting arbitrary data and computation

- Set of all coin commitments
- Old coin serial numbers
- New coin commitments
- ZKP

**Diagram:**

- tx
- rt
- $s_{i}^{\text{old}}$
- $c_{j}^{\text{new}}$
- $\pi$

**Elements:**

- Store arbitrary data
- $\psi$
- $c_{i}^{\text{old}}$
- $s_{i}^{\text{old}}$
- $p_{i}^{\text{old}}$
- $\phi_{b, i}^{\text{old}}$
- $\phi_{d, i}^{\text{old}}$
- $\rho_{i}^{\text{old}}$
- $p_{j}^{\text{new}}$
- $\phi_{b, j}^{\text{new}}$
- $\phi_{d, j}^{\text{new}}$
- $\rho_{j}^{\text{new}}$
- $r_{i}^{\text{old}}$
- $r_{j}^{\text{new}}$

**Formulas:**

- $\phi_{b, i}^{\text{old}}(\cdot) = 1$
- $\phi_{d, i}^{\text{old}}(\cdot) = 1$
- $\phi_{b, j}^{\text{new}}(\cdot) = 1$
Modeling: Records Nano-Kernel

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

libzexe.org