P2P Mixing

A
B
C
D
P2P Mixing

A → B
B → D
C → C
D → A
P2P Mixing

Confirmation
- Peers agree on the output and confirm it

Mix
P2P Mixing

- A
- B
- C
- D

Mix

Confirmation
- Peers agree on the output and confirm it

P2P Trust model
- No mutual trust, no third-party routers
P2P Mixing

Confirmation
• Peers agree on the output and confirm it

P2P Trust model
• No mutual trust, no third-party routers
P2P Mixing

Confirmation
• Peers agree on the output and confirm it

P2P Trust model
• No mutual trust, no third-party routers
• Anonymity set is the set of honest users
P2P Mixing

- A
- B
- C
- D

Mix

Confirmation
- Peers agree on the output and confirm it

P2P Trust model
- No mutual trust, no third-party routers
- Anonymity set is the set of honest users
P2P Mixing

Confirmation
- Peers agree on the output and confirm it

P2P Trust model
- No mutual trust, no third-party routers
- Anonymity set is the set of honest users
- Protocol must terminate in the presence of $f < n - 1$ malicious users
State of the Art (I)
State of the Art (I)

Traditional mixnet run by all peers
State of the Art (I)

Traditional mixnet run by all peers

- Dissent (shuffle protocol) [CCS 2010],
  CoinShuffle [ESORICS 2014]
State of the Art (I)

Traditional mixnet run by all peers

- Dissent (shuffle protocol) [CCS 2010], CoinShuffle [ESORICS 2014]
State of the Art (I)

Traditional mixnet run by all peers

- Dissent (shuffle protocol) [CCS 2010], CoinShuffle [ESORICS 2014]
- $O(n)$ rounds in optimistic case
State of the Art (I)

Traditional mixnet run by all peers
- Dissent (shuffle protocol) [CCS 2010], CoinShuffle [ESORICS 2014]
- $O(n)$ rounds in optimistic case
- $O(nf)$ rounds for $f$ malicious peers
State of the Art (I)

Traditional mixnet run by all peers
- Dissent (shuffle protocol) [CCS 2010], CoinShuffle [ESORICS 2014]
- $O(n)$ rounds in optimistic case
- $O(nf)$ rounds for $f$ malicious peers

Traditional mixnet solution does not scale!
State of the Art (II)
State of the Art (II)

Dining cryptographers’ networks (DC-nets)
State of the Art (II)

Dining cryptographers’ networks (DC-nets)
State of the Art (II)

Dining cryptographers’ networks (DC-nets)

- Hope for $O(1)$ rounds in the optimistic case
State of the Art (II)

Dining cryptographers’ networks (DC-nets)

- Hope for $O(1)$ rounds in the optimistic case
- Easy to disrupt
Dining cryptographers’ networks (DC-nets)

- Hope for $O(1)$ rounds in the optimistic case
- Easy to disrupt
- All approaches to solve disruption problem suffer from drawbacks
State of the Art (II)

Dining cryptographers’ networks (DC-nets)

- Hope for $O(1)$ rounds in the optimistic case
- Easy to disrupt
- All approaches to solve disruption problem suffer from drawbacks
- Golle and Juels [EUROCRYPT 2004]: Honest majority
Dining cryptographers’ networks (DC-nets)

- Hope for $O(1)$ rounds in the optimistic case
- Easy to disrupt
- All approaches to solve disruption problem suffer from drawbacks
- Golle and Juels [EUROCRYPT 2004]: Honest majority

No practical P2P mixing protocol based on DC-nets!
DiceMix

A Practical P2P Mixing Protocol based on DC-nets
DC-net [Chaum 1988]
DC-net [Chaum 1988]
DC-net [Chaum 1988]

\[ 1 + (1 + 1) = 1 \]
DC-net [Chaum 1988]

1 + (1 + 1) = 1

0 + (1 + 0) = 1

1 + (0 + 1) = 0
DC-net [Chaum 1988]

\[
\begin{align*}
1 + (1 + 1) &= 1 \\
0 + (1 + 0) &= 1 \\
1 + (0 + 1) &= 0 \\
1 + 1 + 0 &= 0
\end{align*}
\]
Sending Several Messages [Bos, Boer 1989]

User 1: $m_1$
User 2: $m_2$
User 3: $m_3$
User n: $m_n$

\[\sum_{i=1}^{n} m_i\]
### Sending Several Messages [Bos, Boer 1989]

<table>
<thead>
<tr>
<th>User 1:</th>
<th>User 2:</th>
<th>User 3:</th>
<th>User n:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m_1$</td>
<td>$m_1^2$</td>
<td>$m_1^3$</td>
<td>$m_1^n$</td>
</tr>
<tr>
<td>$m_2$</td>
<td>$m_2^2$</td>
<td>$m_2^3$</td>
<td>$m_2^n$</td>
</tr>
<tr>
<td>$m_3$</td>
<td>$m_3^2$</td>
<td>$m_3^3$</td>
<td>$m_3^n$</td>
</tr>
<tr>
<td>⋮</td>
<td>⋮</td>
<td>⋮</td>
<td>⋮</td>
</tr>
<tr>
<td>$m_n$</td>
<td>$m_n^2$</td>
<td>$m_n^3$</td>
<td>$m_n^n$</td>
</tr>
</tbody>
</table>

\[
\sum_{i=1}^{n} m_i \quad \sum_{i=1}^{n} m_i^2 \quad \sum_{i=1}^{n} m_i^3 \quad \cdots \quad \sum_{i=1}^{n} m_i^n
\]
Newton's identities tell us the coefficients of the polynomial \( \prod_{i=1}^{n} (x - m_i) \).
Sending Several Messages [Bos, Boer 1989]

<table>
<thead>
<tr>
<th>User 1:</th>
<th>$m_1$</th>
<th>$m_1^2$</th>
<th>$m_1^3$</th>
<th>$\ldots$</th>
<th>$m_1^n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 2:</td>
<td>$m_2$</td>
<td>$m_2^2$</td>
<td>$m_2^3$</td>
<td>$\ldots$</td>
<td>$m_2^n$</td>
</tr>
<tr>
<td>User 3:</td>
<td>$m_3$</td>
<td>$m_3^2$</td>
<td>$m_3^3$</td>
<td>$\ldots$</td>
<td>$m_3^n$</td>
</tr>
<tr>
<td>User $n$:</td>
<td>$m_n$</td>
<td>$m_n^2$</td>
<td>$m_n^3$</td>
<td>$\ldots$</td>
<td>$m_n^n$</td>
</tr>
</tbody>
</table>

| $\sum_{i=1}^{n} m_i$ | $\sum_{i=1}^{n} m_i^2$ | $\sum_{i=1}^{n} m_i^3$ | $\ldots$ | $\sum_{i=1}^{n} m_i^n$ |

Newton's identities tell us the coefficients of the polynomial $\prod_{i=1}^{n} (x - m_i)$. → Polynomial factorization recovers the messages.
### Disruption

<table>
<thead>
<tr>
<th>User 1:</th>
<th>$m_1$</th>
<th>$m_1^2$</th>
<th>$m_1^3$</th>
<th>$\cdots$</th>
<th>$m_1^n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 2:</td>
<td>$m_1$</td>
<td>$m_2$</td>
<td>$m_2^3$</td>
<td>$\cdots$</td>
<td>$m_2^n$</td>
</tr>
<tr>
<td>User 3:</td>
<td>$m_3$</td>
<td>$m_3^2$</td>
<td>$m_3^3$</td>
<td>$\cdots$</td>
<td>$m_3^n$</td>
</tr>
<tr>
<td>User $n$:</td>
<td>$m_n$</td>
<td>$m_n^2$</td>
<td>$m_n^3$</td>
<td>$\cdots$</td>
<td>$m_n^n$</td>
</tr>
</tbody>
</table>
### Disruption

<table>
<thead>
<tr>
<th>User 1:</th>
<th>User 2:</th>
<th>User 3:</th>
<th>User n:</th>
</tr>
</thead>
<tbody>
<tr>
<td>![User Icon]</td>
<td>![User Icon]</td>
<td>![User Icon]</td>
<td>![User Icon]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$m_1$</th>
<th>$m_2$</th>
<th>$m_3$</th>
<th>$m_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User 2:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User 3:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User n:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$m_1$</th>
<th>$m_2$</th>
<th>$m_3$</th>
<th>$m_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User 2:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User 3:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User n:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$m_1$</th>
<th>$m_2$</th>
<th>$m_3$</th>
<th>$m_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User 2:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User 3:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User n:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Malicious user stays anonymous!
Handling Disruptions
Handling Disruptions

IN CASE OF DISRUPTION
BREAK ANONYMITY
Flowchart of DiceMix

Generate fresh message
Flowchart of DiceMix

- Generate fresh message
- Key exchange
Flowchart of DiceMix

1. Generate fresh message
2. Key exchange
3. DC-net
Flowchart of DiceMix

1. Generate fresh message
2. Key exchange
3. DC-net
4. Disrupted?

P2P Mixing and Unlinkable Bitcoin Transactions
Flowchart of DiceMix
Flowchart of DiceMix

Generate fresh message ➔ Key exchange ➔ DC-net ➔ Disrupted? ➔ Confirm output

Disrupted?

Y ➔ Reveal secrets

N ➔ Confirm output
Flowchart of DiceMix

1. Generate fresh message
2. Key exchange
3. DC-net
4. Disrupted?
   - Y: Reveal secrets
   - N: Confirm output
5. Y: Discard messages
Flowchart of DiceMix

Generate fresh message → Key exchange → DC-net → Disrupted? → Confirm output

Disrupted? (Y) → Reveal secrets → Discard messages → Find and exclude disruptors

Disrupted? (N) → N → Confirm output
Flowchart of DiceMix

1. Generate fresh message
2. Key exchange
3. DC-net
4. Disrupted? (Y/N)
   - Y: Reveal secrets
   - N: Find and exclude disruptors
5. Confirm output

- Flow: Generate fresh message → Key exchange → DC-net → Disrupted? → Confirm output
- Alternatives: Find and exclude disruptors → Discard messages
Flowchart of DiceMix

1. Generate fresh message
2. Key exchange
3. DC-net
4. Disrupted?
   - N: Confirm output
   - Y: Find and exclude disruptors
5. Discard messages
6. Reveal secrets
7. Missing conf?
Flowchart of DiceMix

Generate fresh message → Key exchange → DC-net → Disrupted? → Confirm output

Find and exclude disruptors → Discard messages → Reveal secrets

Missing conf? → Success!
Flowchart of DiceMix

1. Generate fresh message
2. Key exchange
3. DC-net
4. Disrupted?
   - N: Confirm output
   - Y: Reveal secrets
5. Find and exclude disruptors
6. Discard messages
7. Success!
Freshness Is Necessary

Anonymity

Termination with dishonest majority

Support for fixed messages
Freshness Is Necessary

Anonymity

Termination with dishonest majority

Support for fixed messages
Flowchart of DiceMix

Generate fresh message → Key exchange → DC-net → Disrupted? (N: Confirm output, Y: Discard messages)

- Find and exclude disruptors
- Discard messages
- Reveal secrets

Disrupted? (N: Confirm output, Y: Discard messages)

- Confirm output
- Missing conf? (N: Confirm output, Y: Find and exclude disruptors)

Success!
Flowchart of DiceMix

1. Generate fresh message
2. Key exchange
3. DC-net
4. Disrupted?
5. Confirm output
6. Missing conf?
7. Find and exclude disruptors
8. Discard messages
9. Reveal secrets
10. Success!

Y = Yes, N = No
Communication Rounds (naive)
Communication Rounds (naive)

4 + 4f rounds
Communication Rounds (DiceMix)

Run 1

Run 2

Run 3

(Run 4)
Communication Rounds (DiceMix)

4 + 2f rounds
Performance

![Graph showing performance metrics]

- Wall-clock
- Computation (total)
- Computation (factorization)

Number of nodes

Time [sec]
CoinShuffle++

A P2P Coin Mixing Protocol based on DiceMix
Mixing without a Third Party (CoinJoin)

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A: 1 BTC</td>
<td>C': 1 BTC</td>
</tr>
<tr>
<td>B: 1 BTC</td>
<td>A': 1 BTC</td>
</tr>
<tr>
<td>C: 1 BTC</td>
<td>B': 1 BTC</td>
</tr>
</tbody>
</table>

[CoinJoin, Maxwell 2013]
## Mixing without a Third Party (CoinJoin)

### Input
- A: 1 BTC
- B: 1 BTC
- C: 1 BTC

### Output
- C\': 1 BTC
- A\': 1 BTC
- B\': 1 BTC

### Signature
- \(\text{sig}_A\)
- \(\text{sig}_B\)
- \(\text{sig}_C\)

[CoinJoin, Maxwell 2013]
Mixing without a Third Party (CoinJoin)

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A: 1 BTC</td>
<td>C': 1 BTC</td>
</tr>
<tr>
<td>B: 1 BTC</td>
<td>A': 1 BTC</td>
</tr>
<tr>
<td>C: 1 BTC</td>
<td>B': 1 BTC</td>
</tr>
</tbody>
</table>

Mixed list of fresh addresses

[CoinJoin, Maxwell 2013]
Flowchart of CoinShuffle++

1. Generate fresh Bitcoin address
2. Key exchange
3. DC-net
4. Disrupted?
   - Yes: Find and exclude disruptors
   - No: Proceed to next step
5. Discard messages
6. Reveal secrets
7. Sign CoinJoin transaction
8. Missing sigs?
   - Yes: Reveal secrets
   - No: Proceed to next step
9. Success!
## Comparison of CoinShuffle++ vs. TumbleBit

<table>
<thead>
<tr>
<th></th>
<th>TumbleBit (classic tumbler)</th>
<th>CoinShuffle++</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anonymity set / payment</td>
<td>&gt;&gt; 100</td>
<td>~ 100</td>
</tr>
<tr>
<td>Bandwidth / payment</td>
<td>~ 420 bytes</td>
<td>~ 2250 bytes</td>
</tr>
<tr>
<td>Total running time / payment</td>
<td>&lt; 5 s</td>
<td>&lt; 20 s</td>
</tr>
<tr>
<td>Coordination required</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>
## Comparison of CoinShuffle++ vs. TumbleBit

<table>
<thead>
<tr>
<th></th>
<th>TumbleBit (classic tumbler)</th>
<th>CoinShuffle++</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Anonymity set / payment</strong></td>
<td>&gt;&gt; 100</td>
<td>~ 100</td>
</tr>
<tr>
<td><strong>Bandwidth / payment</strong></td>
<td>~ 420 bytes</td>
<td>~ 2250 bytes</td>
</tr>
<tr>
<td><strong>Total running time / payment</strong></td>
<td>&lt; 5 s</td>
<td>&lt; 20 s</td>
</tr>
<tr>
<td><strong>Coordination required</strong></td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td><strong>Sequential blocks / payment</strong></td>
<td>2 + 1</td>
<td>1 + 1</td>
</tr>
<tr>
<td><strong>Input-output pairs / payment</strong></td>
<td>4 + 1</td>
<td>1 + 1</td>
</tr>
<tr>
<td><strong>Centralization</strong></td>
<td>dedicated tumbler</td>
<td>P2P with bulletin board</td>
</tr>
<tr>
<td><strong>Collateral required</strong></td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

- **off-chain** vs **on-chain**
## Comparison of CoinShuffle++ vs. TumbleBit

<table>
<thead>
<tr>
<th></th>
<th>TumbleBit (classic tumbler)</th>
<th>CoinShuffle++</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Anonymity set / payment</strong></td>
<td>&gt;&gt; 100</td>
<td>~ 100</td>
</tr>
<tr>
<td><strong>Bandwidth / payment</strong></td>
<td>~ 420 bytes</td>
<td>~ 2250 bytes</td>
</tr>
<tr>
<td><strong>Total running time / payment</strong></td>
<td>&lt; 5 s</td>
<td>&lt; 20 s</td>
</tr>
<tr>
<td><strong>Coordination required</strong></td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td><strong>Sequential blocks / payment</strong></td>
<td>2 + 1</td>
<td>1 + 1</td>
</tr>
<tr>
<td><strong>Input-output pairs / payment</strong></td>
<td>4 + 1</td>
<td>1 + 1</td>
</tr>
<tr>
<td><strong>Centralization</strong></td>
<td>dedicated tumbler</td>
<td>P2P with bulletin board</td>
</tr>
<tr>
<td><strong>Collateral required</strong></td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td><strong>DoS / Sybil protection</strong></td>
<td>fees</td>
<td>performance penalty / fees</td>
</tr>
<tr>
<td><strong>Confidential Transactions</strong></td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>
More Shuffling

Shameless Plugs
ValueShuffle

ValueShuffle \equiv \text{CoinShuffle++} + \text{Confidential Transactions}
ValueShuffle

- Hides amounts in transactions and provides anonymity
ValueShuffle

- Hides amounts in transactions and provides anonymity
- **Users with different amounts of money can mix!**
ValueShuffle

- Hides amounts in transactions and provides anonymity
- **Users with different amounts of money can mix!**
- User can mix and pay simultaneously in one transaction
ValueShuffle

- Hides amounts in transactions and provides anonymity
- **Users with different amounts of money can mix!**
- User can mix and pay simultaneously in one transaction
- Accepted at Bitcoin Workshop 2017
ValueShuffle

ValueShuffle \equiv \quad \text{CoinShuffle++} \quad \text{Confidential Transactions}

- Hides amounts in transactions and provides anonymity
- **Users with different amounts of money can mix!**
- User can mix and pay simultaneously in one transaction
- Accepted at Bitcoin Workshop 2017
- See our poster tonight!
Money Mixing in Credit Networks

- Idea similar to CoinJoin, but there is no CoinJoin transaction
- Challenge: Simulate the CoinJoin via a shared account
PathShuffle

Money Mixing in Credit Networks

- Idea similar to CoinJoin, but there is no CoinJoin transaction
- Challenge: Simulate the CoinJoin via a shared account
Money Mixing in Credit Networks

- Idea similar to CoinJoin, but there is no CoinJoin transaction
- Challenge: Simulate the CoinJoin via a shared account

Accepted by PoPETs 2017
Take-Home Message

• DC-nets are practical
  • No honest majority necessary
  • Only simple crypto, simple protocol
  • Only 4 + 2f rounds

• P2P coin mixing is practical
  • No central party necessary
  • CoinShuffle++ is an efficient solution

Work in progress:
https://github.com/real-or-random/python-dicemix
Backup Slides
Freshness Is Necessary

Anonymity

Termination with dishonest majority

Support for fixed messages
Idea of Attack on Anonymity

\[ m_1 \rightarrow m_2 \]
\[ m_2 \rightarrow m_4 \]
\[ m_3 \rightarrow m_3 \]
\[ m_4 \rightarrow m_1 \]
Idea of Attack on Anonymity

\[ m_1 \rightarrow m_2 \rightarrow m_4 \rightarrow m_3 \rightarrow m_1 \]
Idea of Attack on Anonymity

\[ m_1 \xrightarrow{\text{}} m_2 \xrightarrow{\text{}} m_3 \xrightarrow{\text{}} m_4 \]
Idea of Attack on Anonymity
Idea of Attack on Anonymity

\[ m_1 \rightarrow m_2 \]
\[ m_2 \rightarrow m_4 \]
\[ m_3 \rightarrow m_3 \]
\[ m_4 \rightarrow m_1 \]
Idea of Attack on Anonymity

\[ m_1 \rightarrow m_2 \rightarrow m_4 \rightarrow m_3 \rightarrow m_1 \]
Idea of Attack on Anonymity

\[ m_1 \rightarrow m_2 \leftarrow m_3 \rightarrow m_4 \]
Idea of Attack on Anonymity

\[ m_1 \xrightarrow{} m_2 \]
\[ m_2 \xrightarrow{} m_4 \]
\[ m_3 \xrightarrow{} m_3 \]
\[ m_4 \xrightarrow{} m_1 \]
Idea of Attack on Anonymity

\[ m_1 \xrightarrow{} m_2 \xrightarrow{} m_4 \xrightarrow{} m_3 \xrightarrow{} m_1 \]
### Communication Rounds (DiceMix)

<table>
<thead>
<tr>
<th></th>
<th>KE</th>
<th>DC</th>
<th>?</th>
<th>SK</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Run 1</strong></td>
<td>KE</td>
<td>DC</td>
<td>?</td>
<td>SK</td>
</tr>
<tr>
<td><strong>Run 2</strong></td>
<td>KE</td>
<td>DC</td>
<td>?</td>
<td>RV</td>
</tr>
<tr>
<td><strong>Run 3</strong></td>
<td>RV</td>
<td>RV</td>
<td>RV</td>
<td>RV</td>
</tr>
<tr>
<td><strong>(Run 4)</strong></td>
<td>RV</td>
<td>RV</td>
<td>RV</td>
<td>RV</td>
</tr>
</tbody>
</table>
Communication Rounds (DiceMix)

<table>
<thead>
<tr>
<th>Run 1</th>
<th>KE</th>
<th>DC</th>
<th>?</th>
<th>SK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run 2</td>
<td>KE</td>
<td>DC</td>
<td>?</td>
<td>RV</td>
</tr>
<tr>
<td>Run 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Run 4)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Communication Rounds (DiceMix)

<table>
<thead>
<tr>
<th>Run 1</th>
<th>KE</th>
<th>DC</th>
<th>?</th>
<th>SK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run 2</td>
<td>KE</td>
<td>DC</td>
<td>?</td>
<td>RV</td>
</tr>
<tr>
<td>Run 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Run 4)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- KE: Key Encryption
- DC: Data Consumption
- SK: Secret Key
- RV: Random Value
# Communication Rounds (DiceMix)

<table>
<thead>
<tr>
<th>Run 1</th>
<th>KE</th>
<th>DC</th>
<th>?</th>
<th>SK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run 2</td>
<td>KE</td>
<td>DC</td>
<td>?</td>
<td>RV</td>
</tr>
<tr>
<td>Run 3</td>
<td>RV</td>
<td>RV</td>
<td>RV</td>
<td>RV</td>
</tr>
<tr>
<td>(Run 4)</td>
<td>RV</td>
<td>RV</td>
<td>RV</td>
<td>RV</td>
</tr>
</tbody>
</table>
### Communication Rounds (DiceMix)

<table>
<thead>
<tr>
<th>Run 1</th>
<th>KE</th>
<th>DC</th>
<th>?</th>
<th>SK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run 2</td>
<td>KE</td>
<td>DC</td>
<td>?</td>
<td>RV</td>
</tr>
<tr>
<td>Run 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Run 4)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**4 + 2f rounds**
## Communication Rounds (DiceMix)

<table>
<thead>
<tr>
<th>Run</th>
<th>KE</th>
<th>CM</th>
<th>DC</th>
<th>SK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Run 2</td>
<td></td>
<td></td>
<td>KE</td>
<td>CM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DC</td>
</tr>
<tr>
<td>Run 3</td>
<td></td>
<td></td>
<td></td>
<td>RV</td>
</tr>
</tbody>
</table>

(Run 4)

P2P Mixing and Unlinkable Bitcoin Transactions

Tim Ruffing - @real_or_random
## Communication Rounds (DiceMix)

<table>
<thead>
<tr>
<th>Run 1</th>
<th>KE</th>
<th>CM</th>
<th>DC</th>
<th>SK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run 2</td>
<td>KE</td>
<td>CM</td>
<td>DC</td>
<td>RV</td>
</tr>
<tr>
<td>Run 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Run 4)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Communication Rounds (DiceMix)

<table>
<thead>
<tr>
<th>Run 1</th>
<th>Run 2</th>
<th>Run 3</th>
<th>(Run 4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KE</td>
<td>KE</td>
<td>KE</td>
<td>KE</td>
</tr>
<tr>
<td>CM</td>
<td>CM</td>
<td>CM</td>
<td>CM</td>
</tr>
<tr>
<td>DC</td>
<td>DC</td>
<td>RV</td>
<td>RV</td>
</tr>
<tr>
<td>SK</td>
<td>SK</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**P2P Mixing and Unlinkable Bitcoin Transactions**

**Tim Ruffing - @real_or_random**
Communication Rounds (DiceMix)

Run 1
- KE
- CM
- DC
- SK

Run 2
- KE
- CM
- DC
- RV

Run 3
- RV
- RV
- RV
- RV

(Run 4)
**Communication Rounds (DiceMix)**

<table>
<thead>
<tr>
<th>Run 1</th>
<th>Run 2</th>
<th>Run 3</th>
<th>(Run 4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KE</td>
<td>CM</td>
<td>DC</td>
<td>SK</td>
</tr>
<tr>
<td>KE</td>
<td>CM</td>
<td>DC</td>
<td>RV</td>
</tr>
</tbody>
</table>

4 + 2f rounds
Transaction Linkability

A Threat to Privacy
Linkability and Deanonymization Attacks

[Meiklejohn et al. 2013]
Linkability and Deanonymization Attacks

P2P Mixing and Unlinkable Bitcoin Transactions

Tim Ruffing – @real_or_random
Linkability and Deanonymization Attacks

Bitlodine [Spagnuolo, Maggi, Zanero 2013]
Linkability and Deanonymization Attacks

Bitlodine [Spagnuolo, Maggi, Zanero 2013]

Multi-input
Shadow

BitCluster
The Bitcoin De-Anonymizer

P2P Mixing and Unlinkable Bitcoin Transactions

Tim Ruffing – @real_or_random
Linkability and Deanonymization Attacks

**REACTOR**

**Start from anywhere** — Have a specific customer that you are interested in? Or a ransom note with a Bitcoin address? Have some plain text that you don’t know if it contains Bitcoin references? Paste it in to the tool and it will automatically find connected Bitcoin wallets.

**Interactive investigation tool** — Annotate your findings and keep notes on what led you to those conclusions. Identify reappearing offenders and share data with other people in your organization.

**Visualize** — Annotated data is displayed with charts and a graphing space to spot connections, explore different hypotheses and gain quick insights.

Bitlodine [Spagnuolo, Maggi, Zanero 2013]
Performance

• We use an untrusted bulletin board, e.g., IRC server, but just for communication.

• CoinShuffle++ terminates in $4 + 2f$ rounds with $f$ disruptive users

• $< 10$ seconds to create CoinJoin transaction with 50 honest users (unoptimized)

• old CoinShuffle: about 3 min

• Work in progress:
  https://github.com/real-or-random/python-dicemix
DC-nets in Practice
DC-nets in Practice

• Key exchange to establish shared keys
DC-nets in Practice

• Key exchange to establish shared keys
• Send bitstrings instead of single bits
DC-nets in Practice

- Key exchange to establish shared keys
- Send bitstrings instead of single bits
- DC-nets computes sum, but should compute set of messages
DC-nets in Practice

- Key exchange to establish shared keys
- Send bitstrings instead of single bits
- DC-nets computes sum, but should compute set of messages
  - Often: Use „slots“ and perform slot reservation

![Example bitstrings and key exchange]

```
00000 10110 00000
00110 00000 00000
00000 00000 11001
```