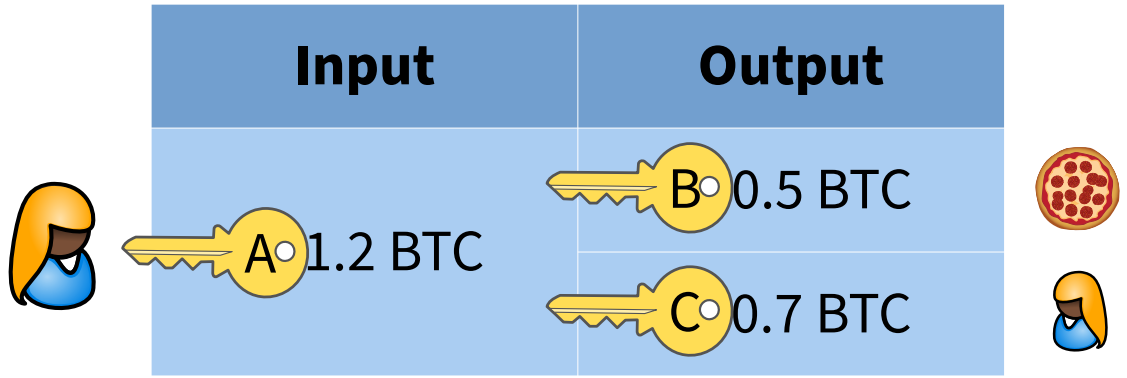
Peer-to-Peer Payment System

Peer-to-Peer Payment System

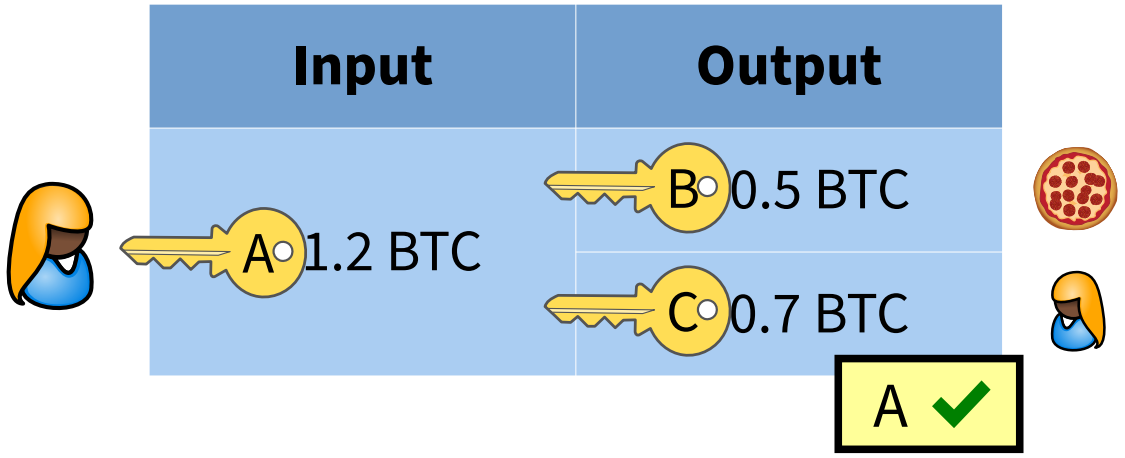
Peer-to-Peer Payment System

Input	Output
A: 1.2 BTC	B: 0.5 BTC
	C: 0.7 BTC

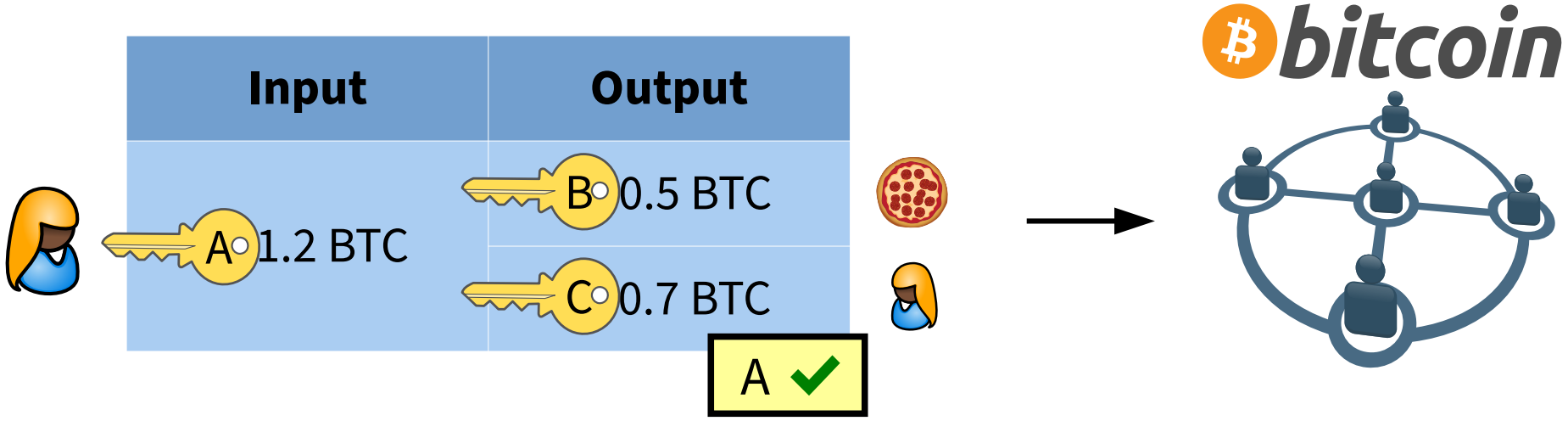
Peer-to-Peer Payment System



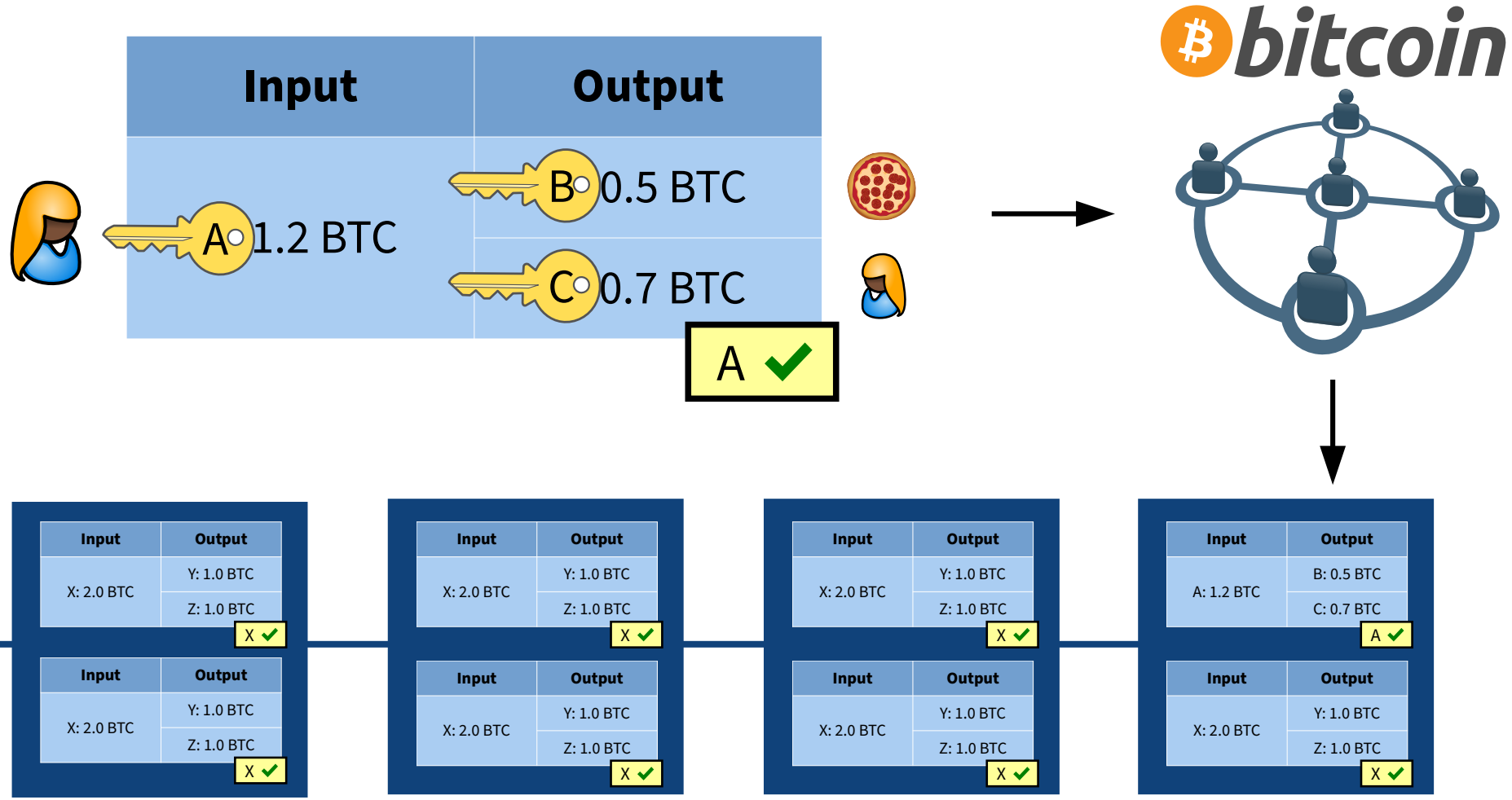
Peer-to-Peer Payment System



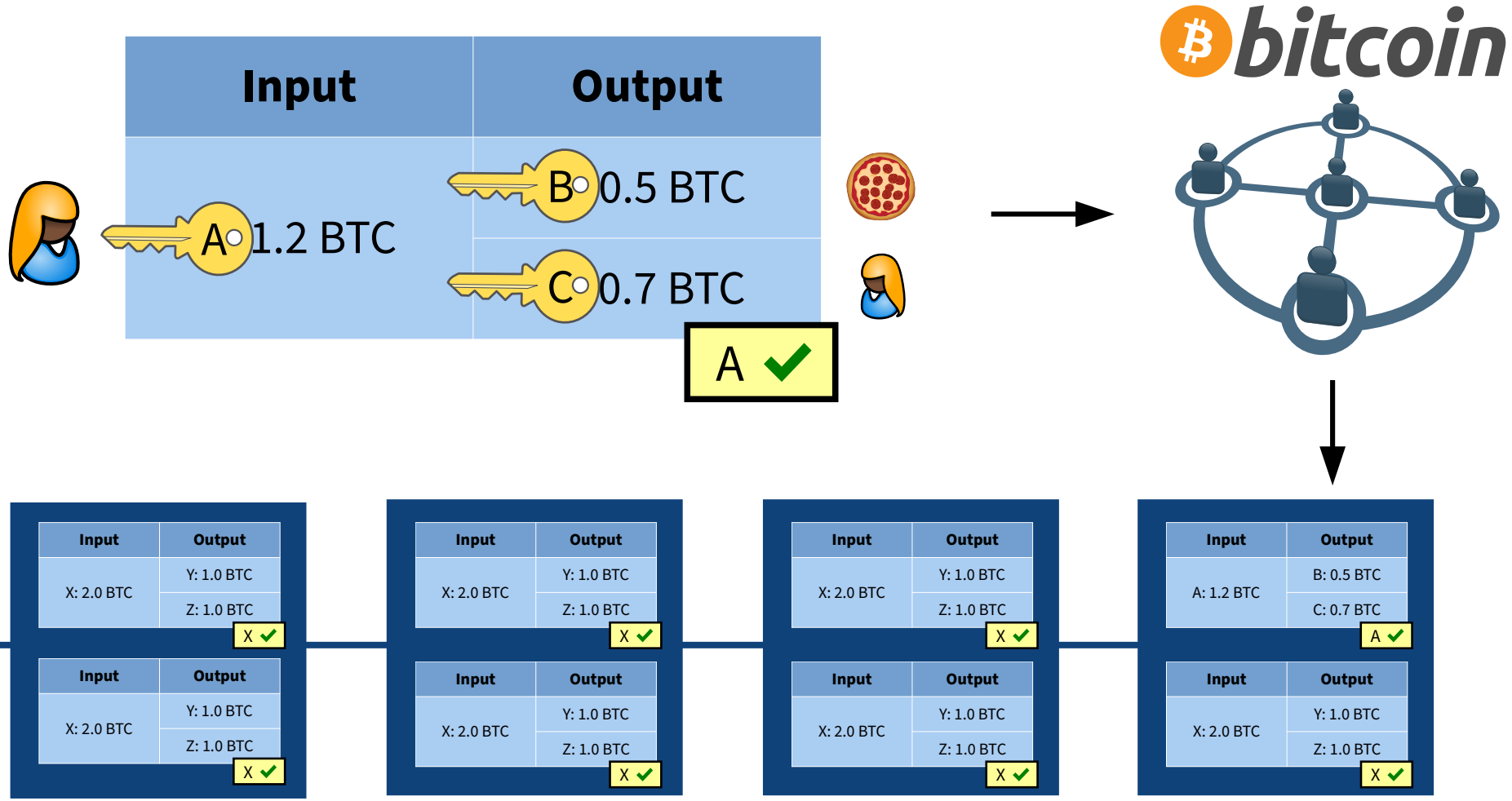
Peer-to-Peer Payment System



Peer-to-Peer Payment System



Peer-to-Peer Payment System



blockchain = **public** ledger of transactions

Main Privacy Issues (Transaction Layer)

Main Privacy Issues (Transaction Layer)

- Transacted amounts are public

Main Privacy Issues (Transaction Layer)

- Transacted amounts are public

Input	Output
A: 1.2 BTC	B: 0.5 BTC
	C: 0.7 BTC

Main Privacy Issues (Transaction Layer)

- Transacted amounts are public
- Addresses (pseudonyms) can be linked

Input	Output
A: 1.2 BTC	B: 0.5 BTC
	C: 0.7 BTC

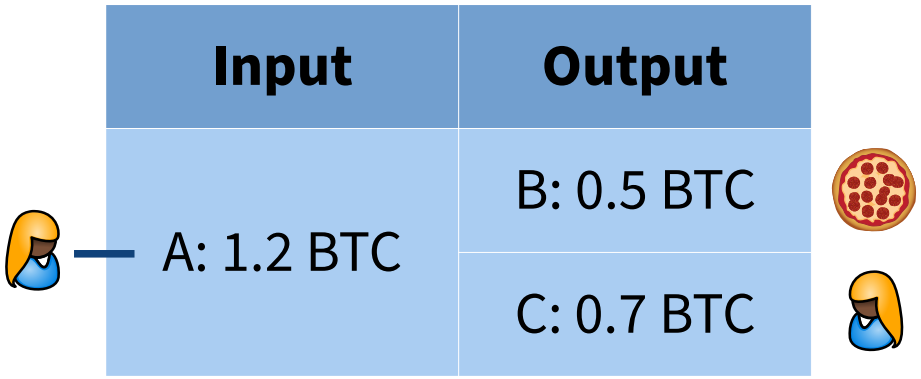
Main Privacy Issues (Transaction Layer)

- Transacted amounts are public
- Addresses (pseudonyms) can be linked

Input	Output
A: 1.2 BTC	B: 0.5 BTC
	C: 0.7 BTC

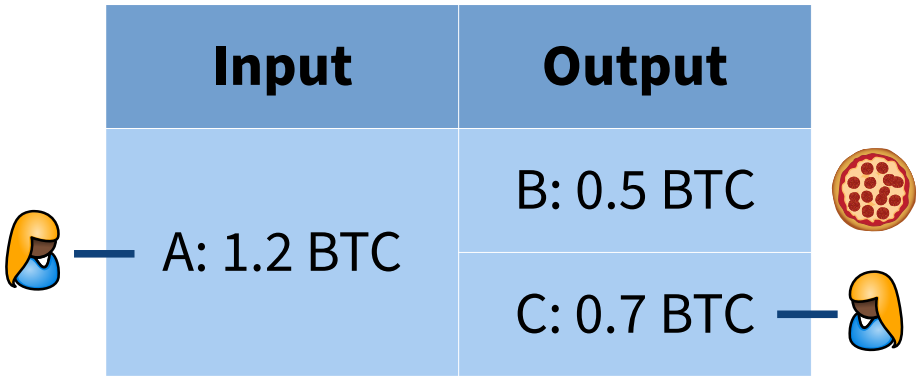
Main Privacy Issues (Transaction Layer)

- Transacted amounts are public
- Addresses (pseudonyms) can be linked



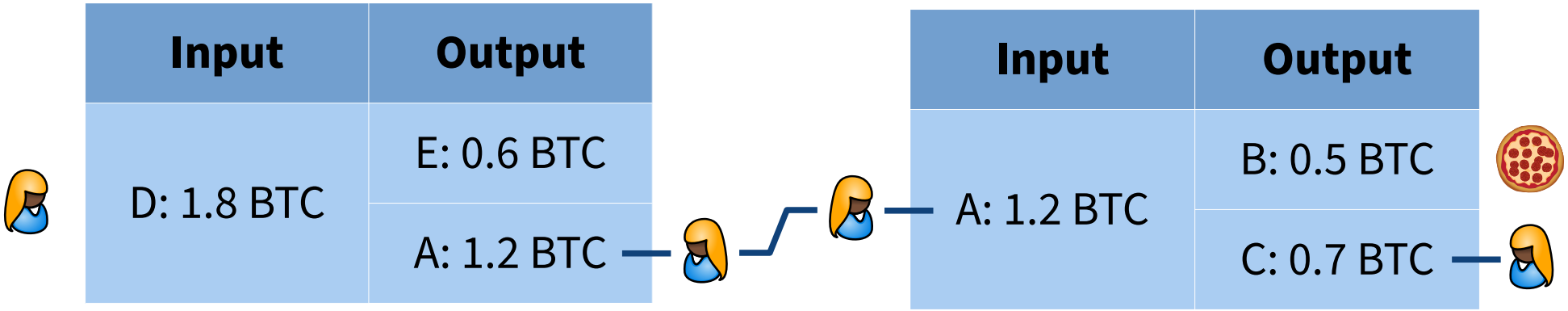
Main Privacy Issues (Transaction Layer)

- Transacted amounts are public
- Addresses (pseudonyms) can be linked



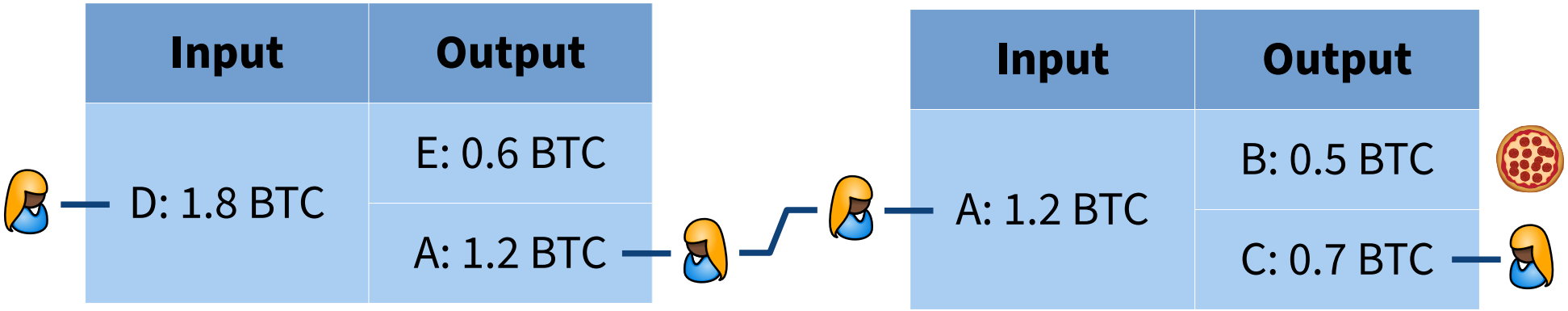
Main Privacy Issues (Transaction Layer)

- Transacted amounts are public
- Addresses (pseudonyms) can be linked

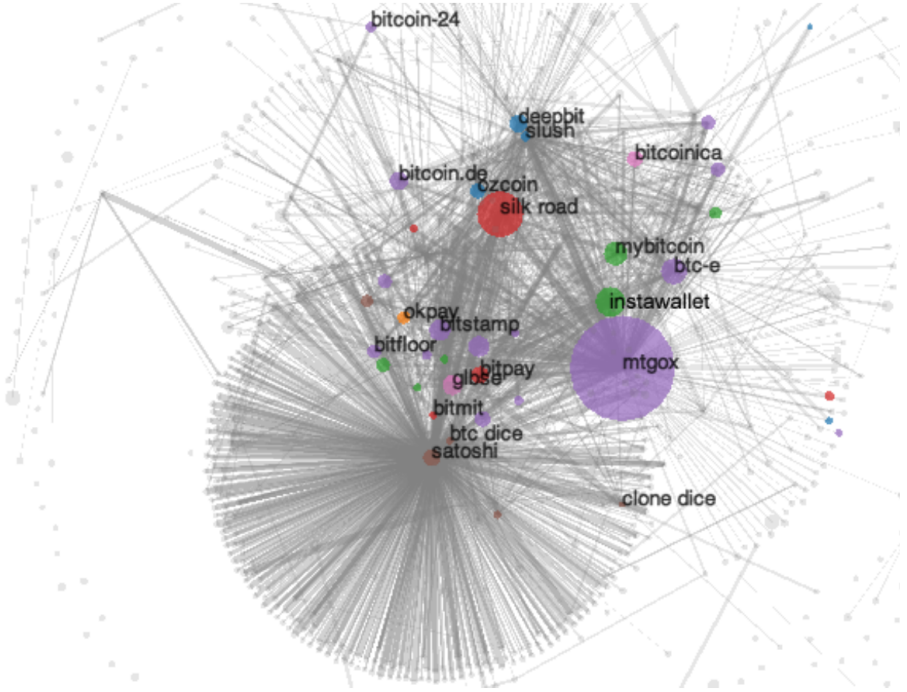


Main Privacy Issues (Transaction Layer)

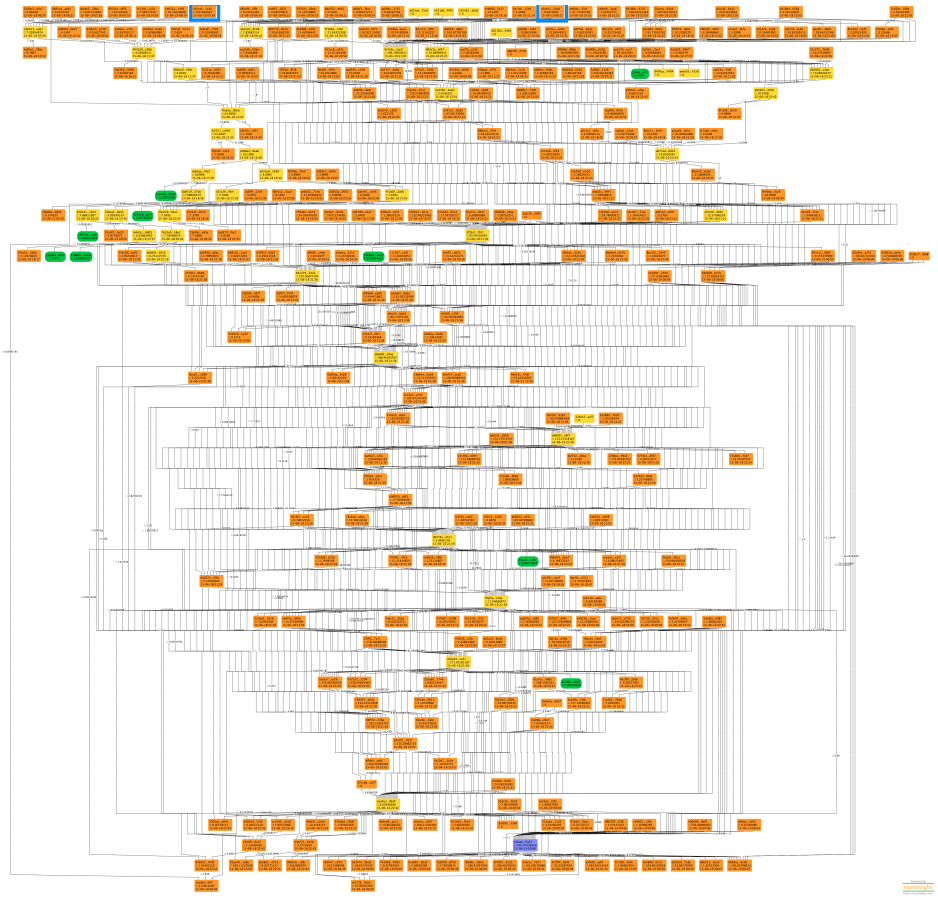
- Transacted amounts are public
- Addresses (pseudonyms) can be linked



Transaction Graph Analysis

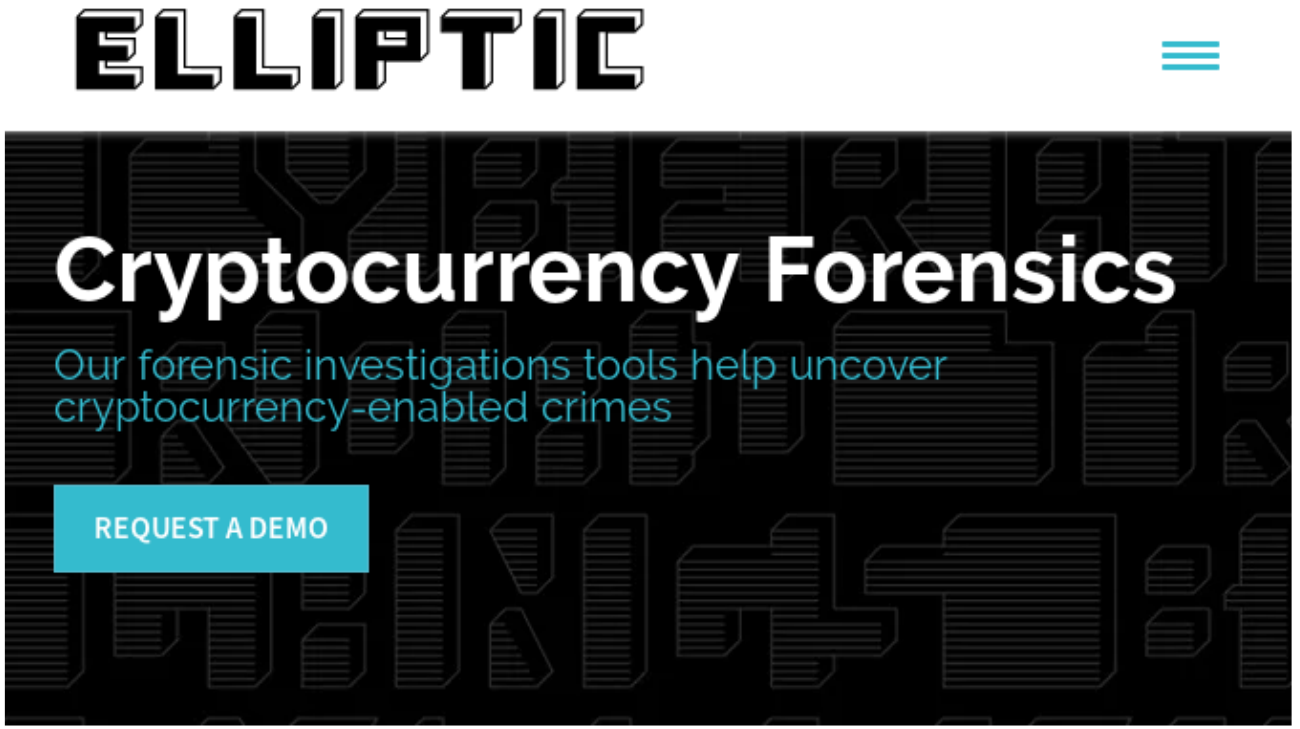


[Meiklejohn et al. 2013]

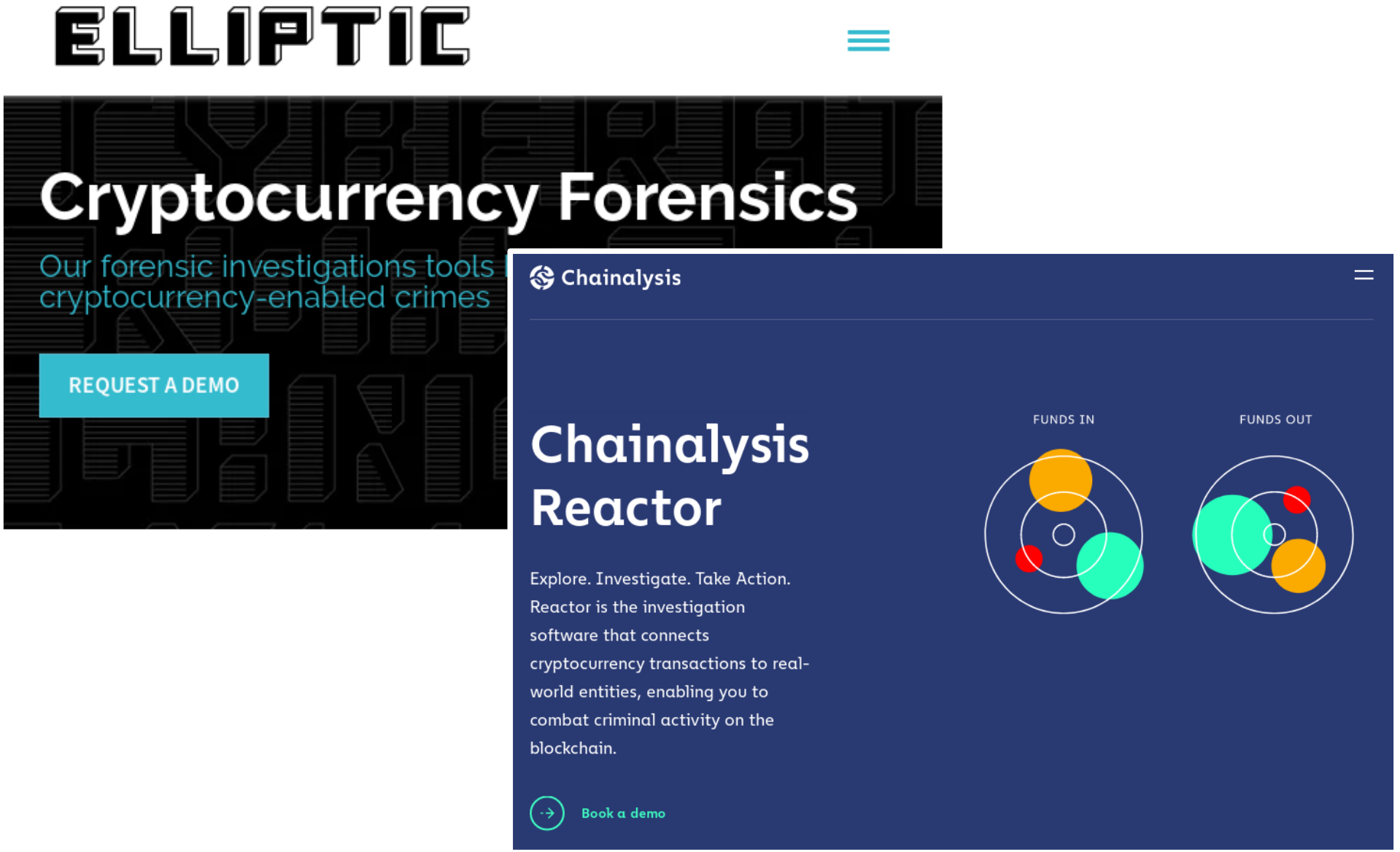


BitIodine [Spagnuolo, Maggi, Zanero 2013]

Transaction Graph Analysis

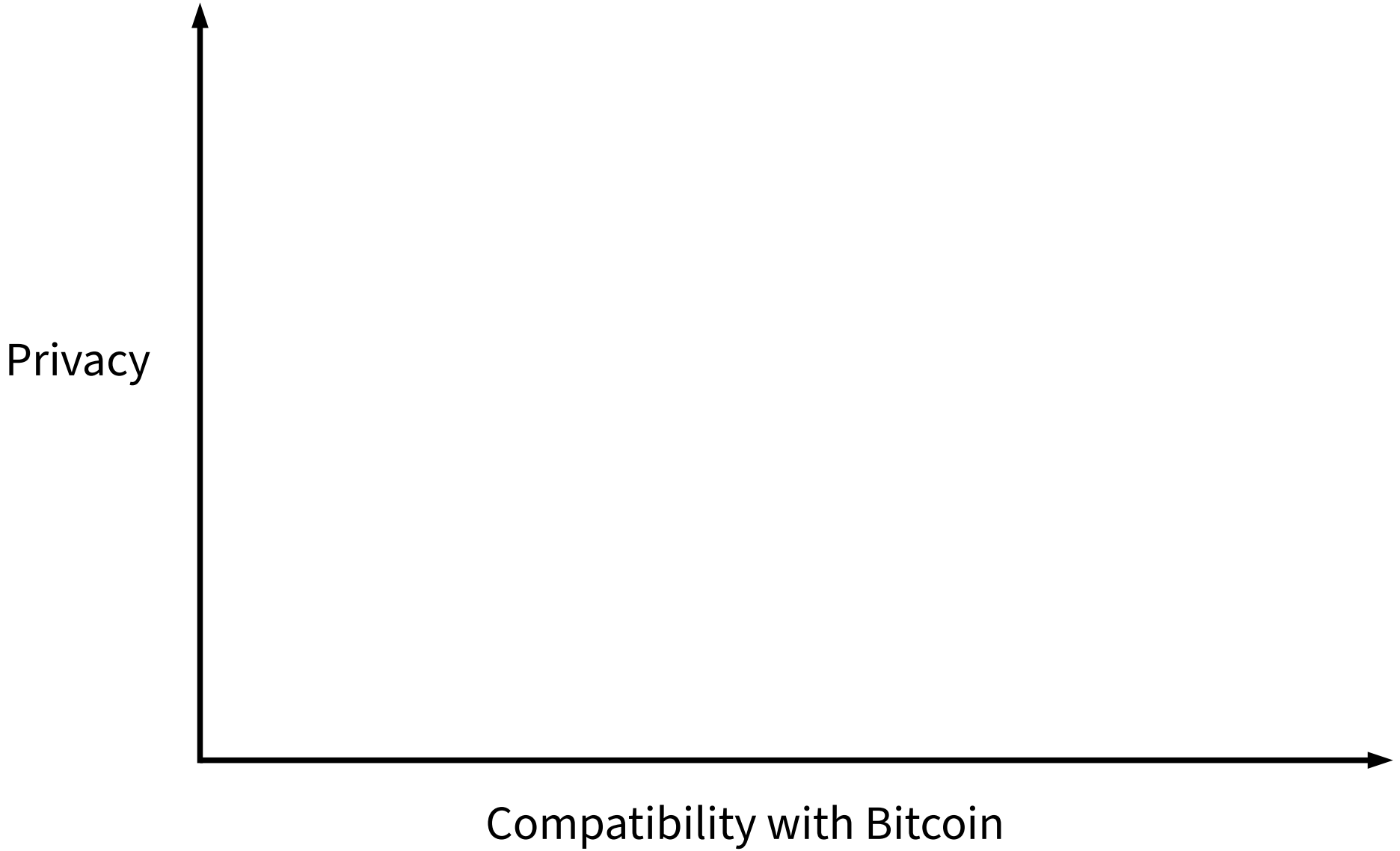


Transaction Graph Analysis

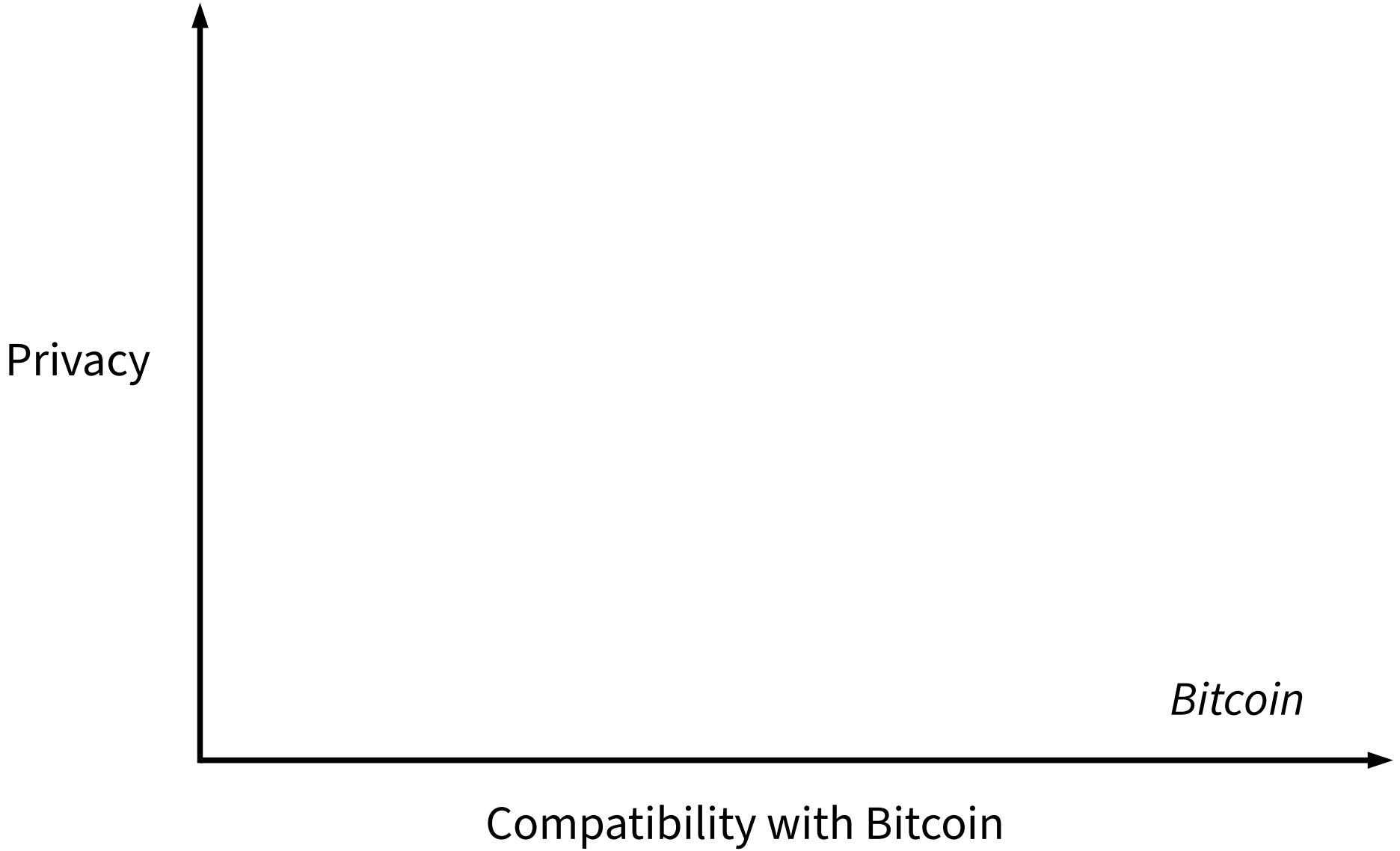


Improving Privacy

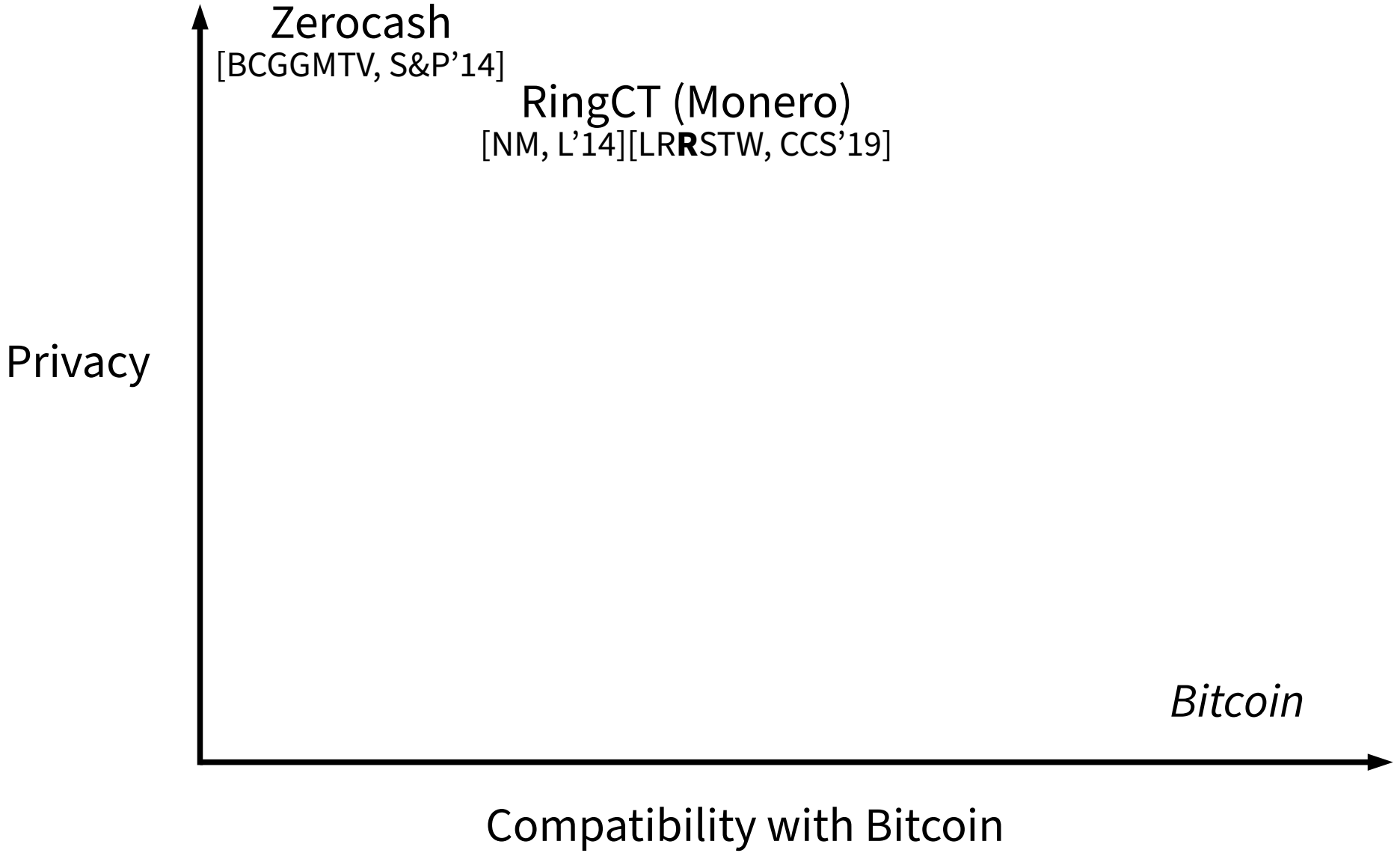
Big Picture of Privacy Technologies



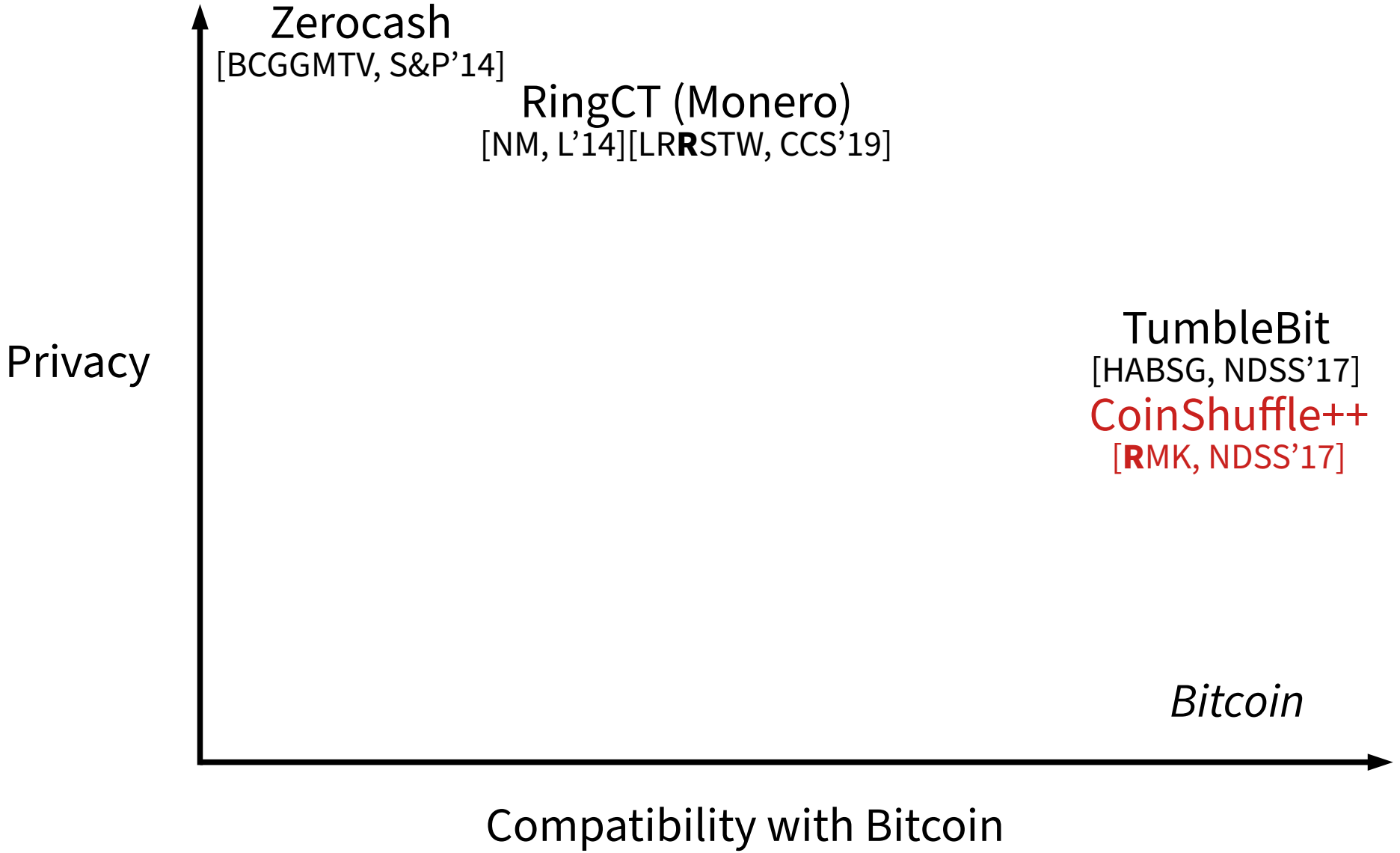
Big Picture of Privacy Technologies



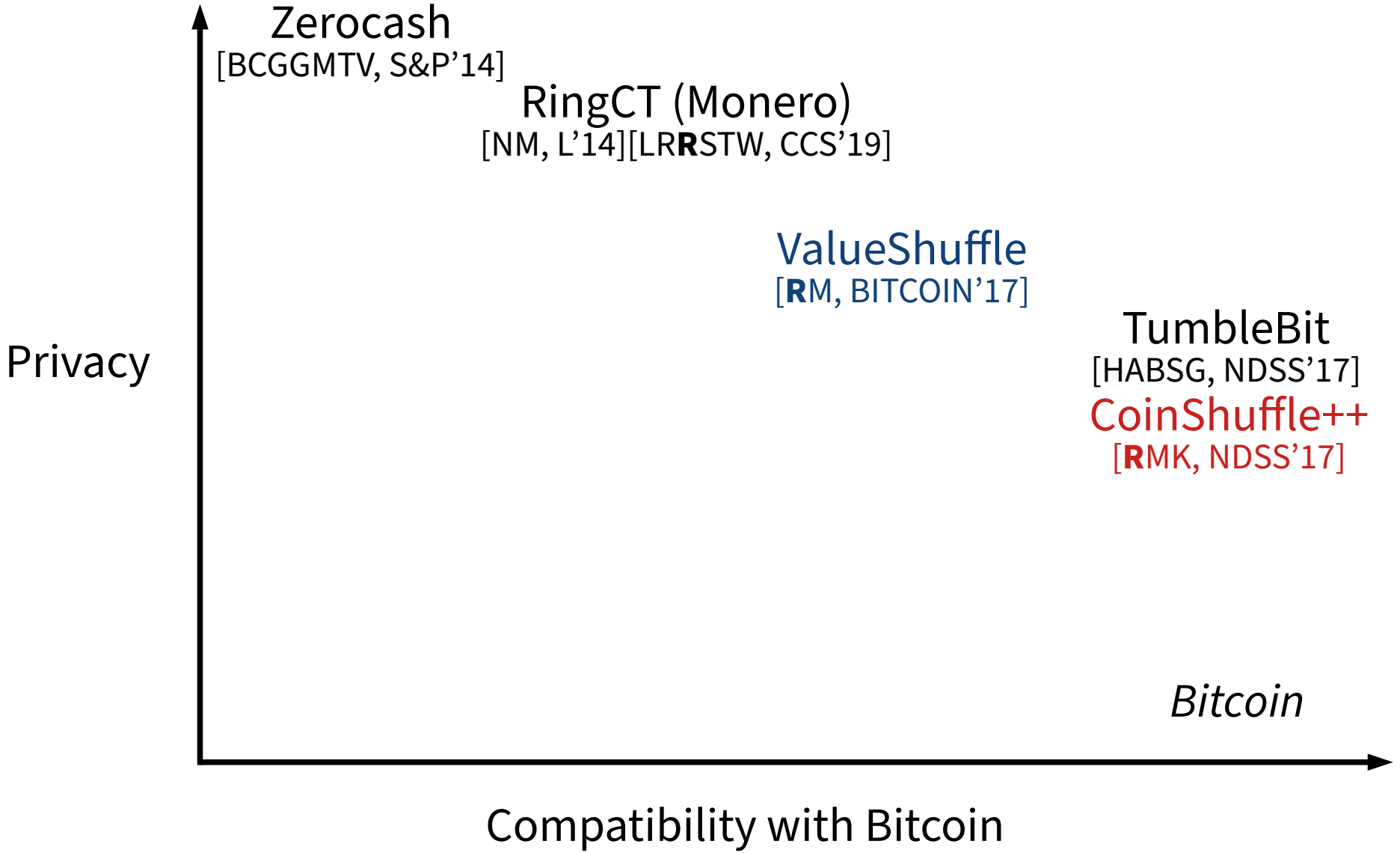
Big Picture of Privacy Technologies



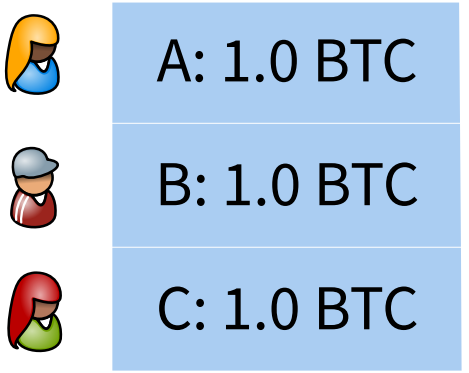
Big Picture of Privacy Technologies



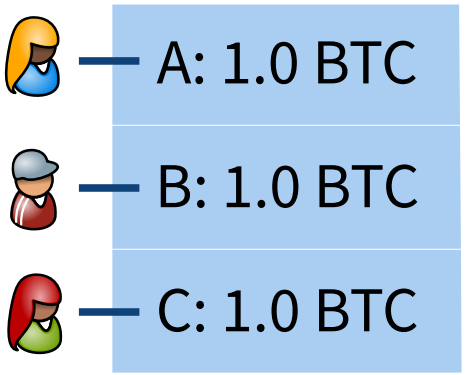
Big Picture of Privacy Technologies



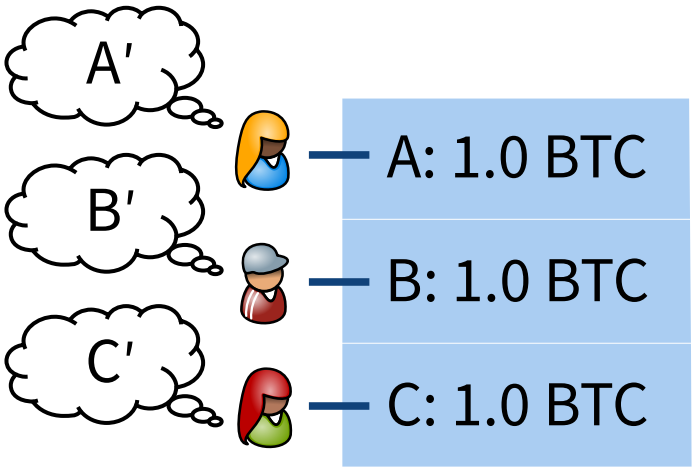
Coin Mixing



Coin Mixing

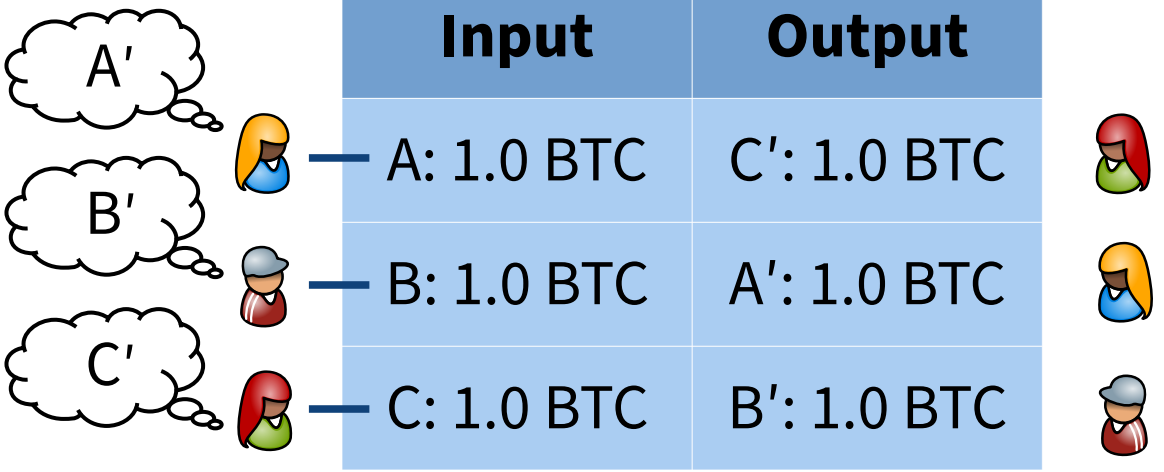


Coin Mixing



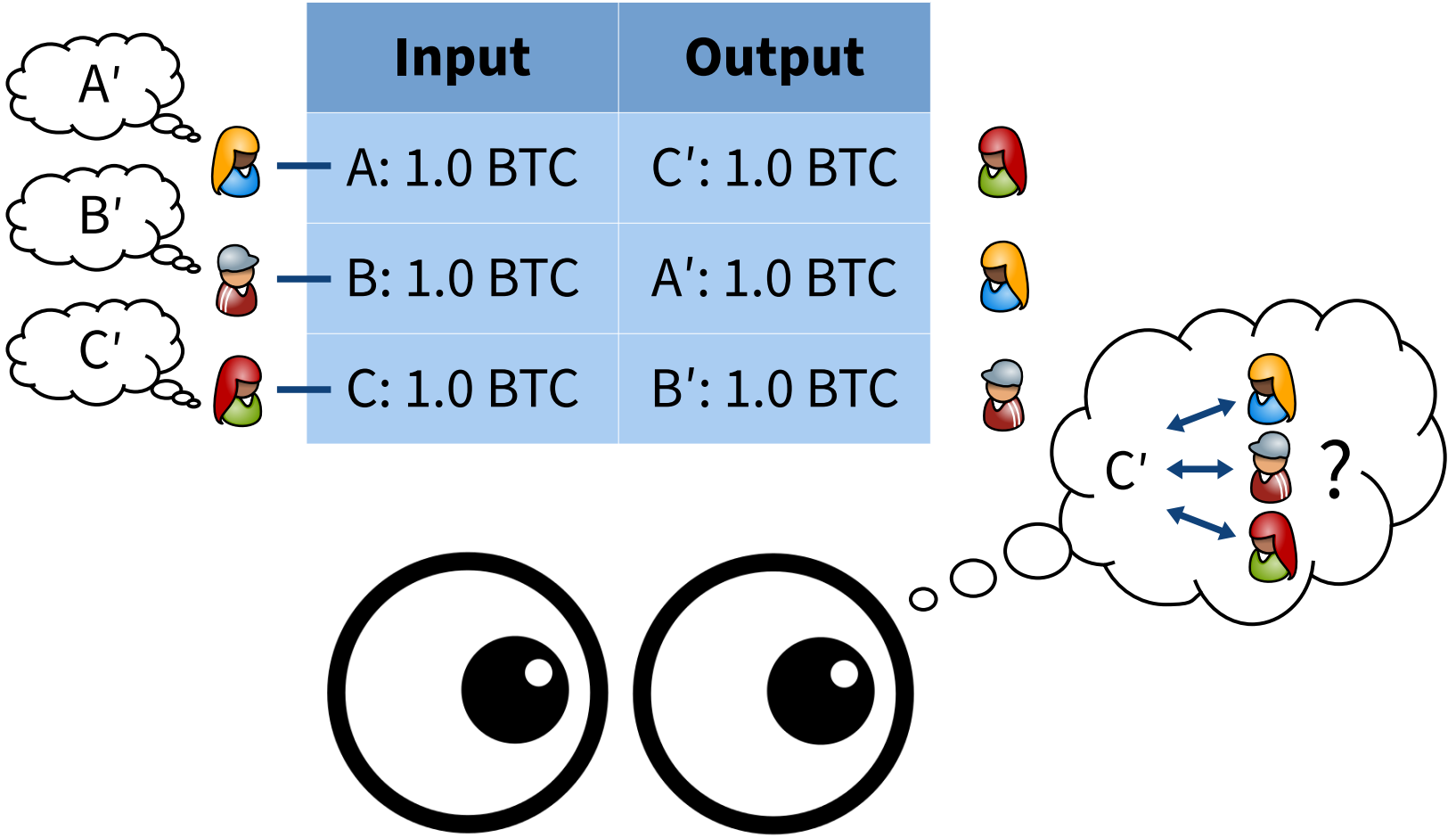
Coin Mixing

Multi-input multi-output transaction



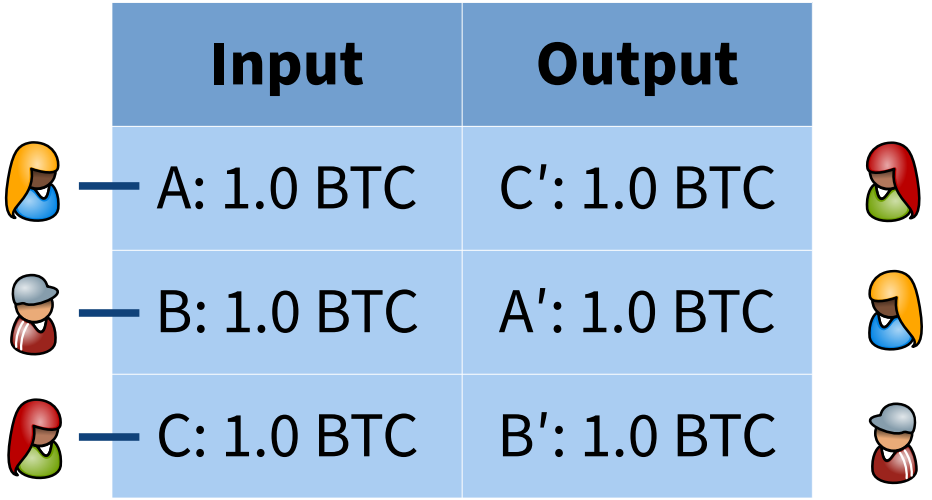
Coin Mixing

Multi-input multi-output transaction



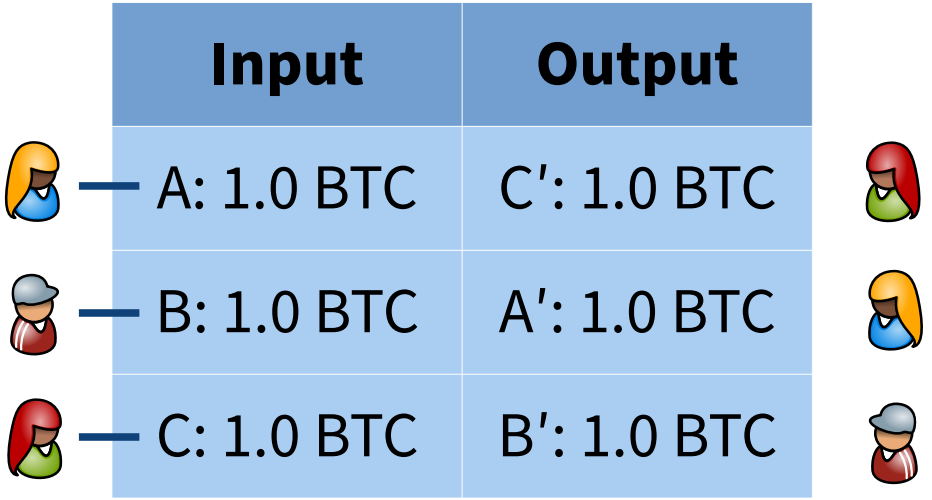
Coin Mixing

Multi-input multi-output transaction



Coin Mixing

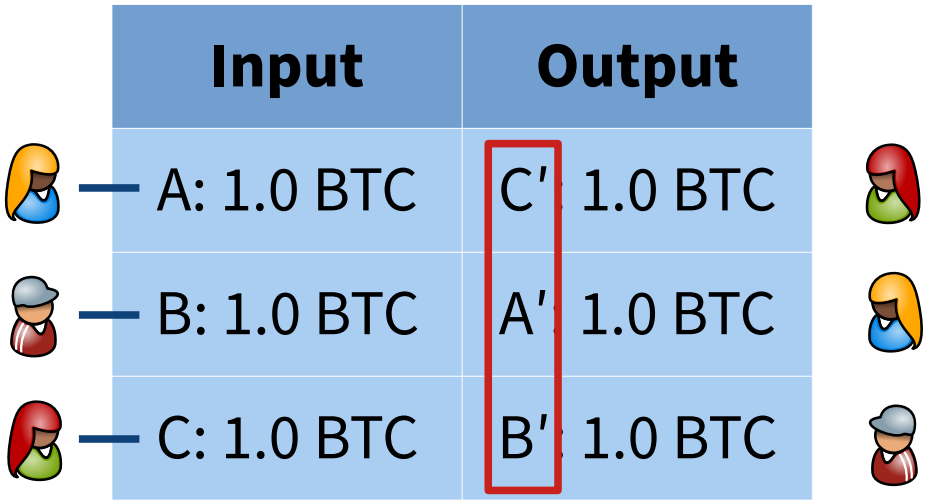
Multi-input multi-output transaction



Fully compatible with Bitcoin

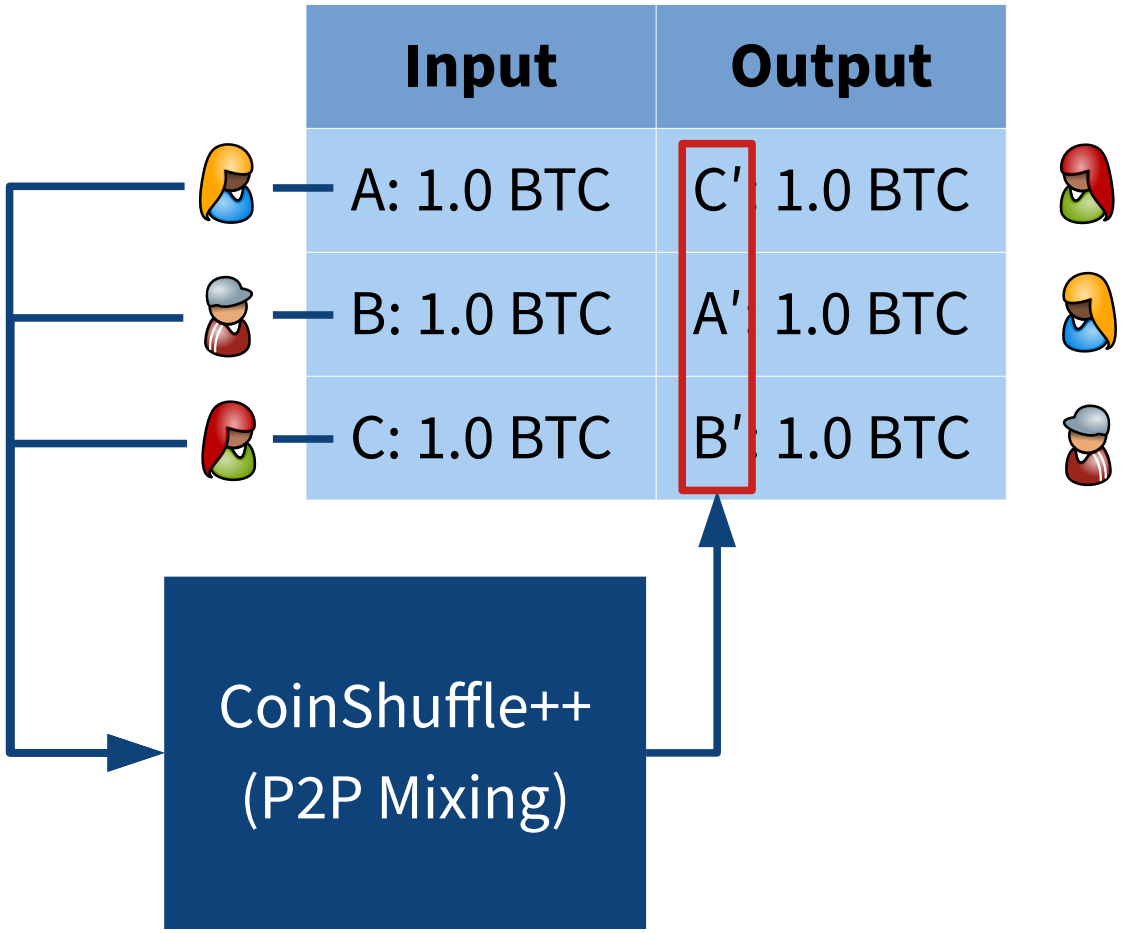
Coin Mixing

Multi-input multi-output transaction

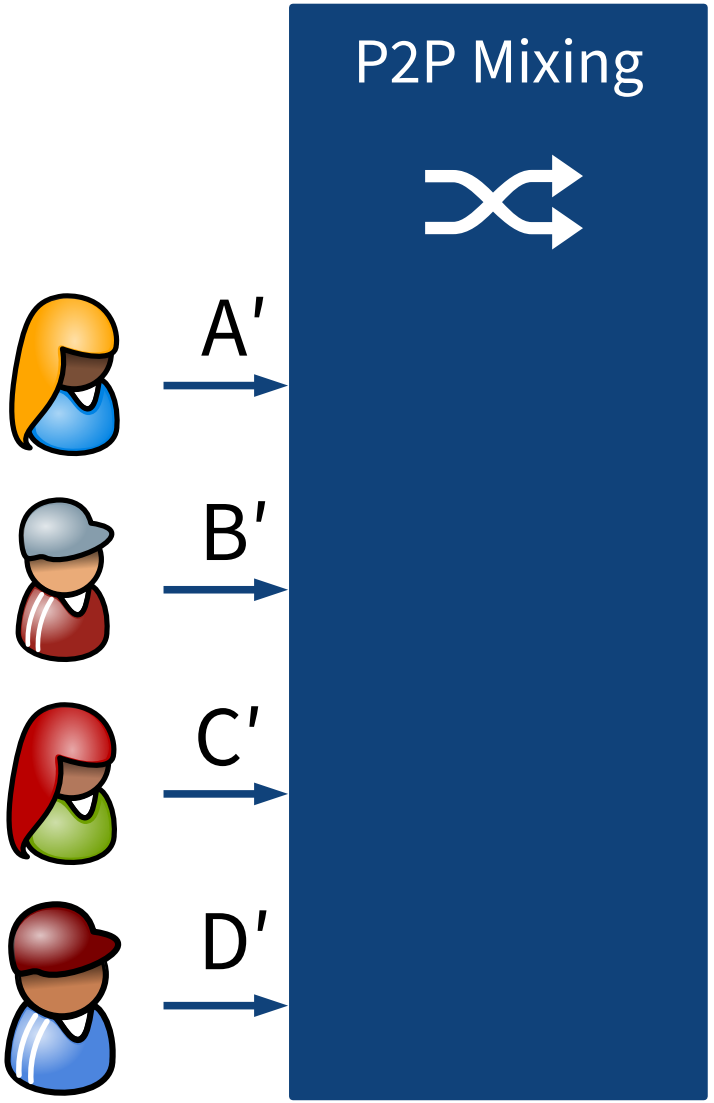


Coin Mixing

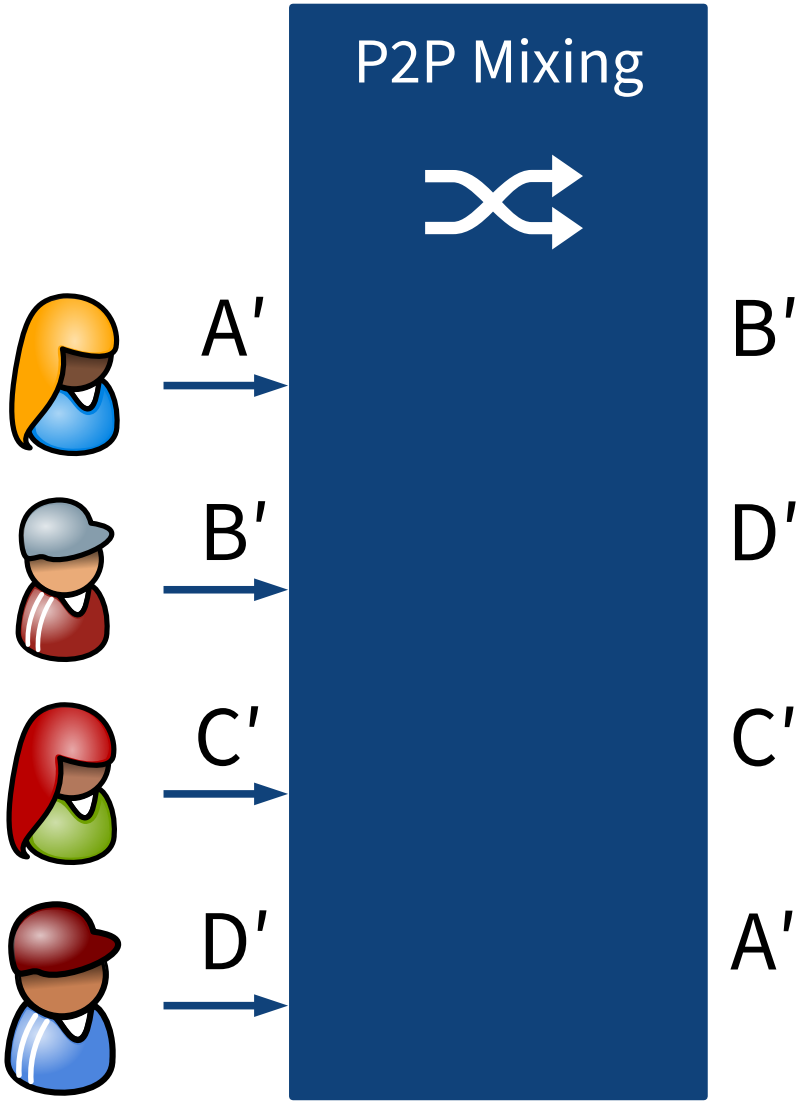
Multi-input multi-output transaction



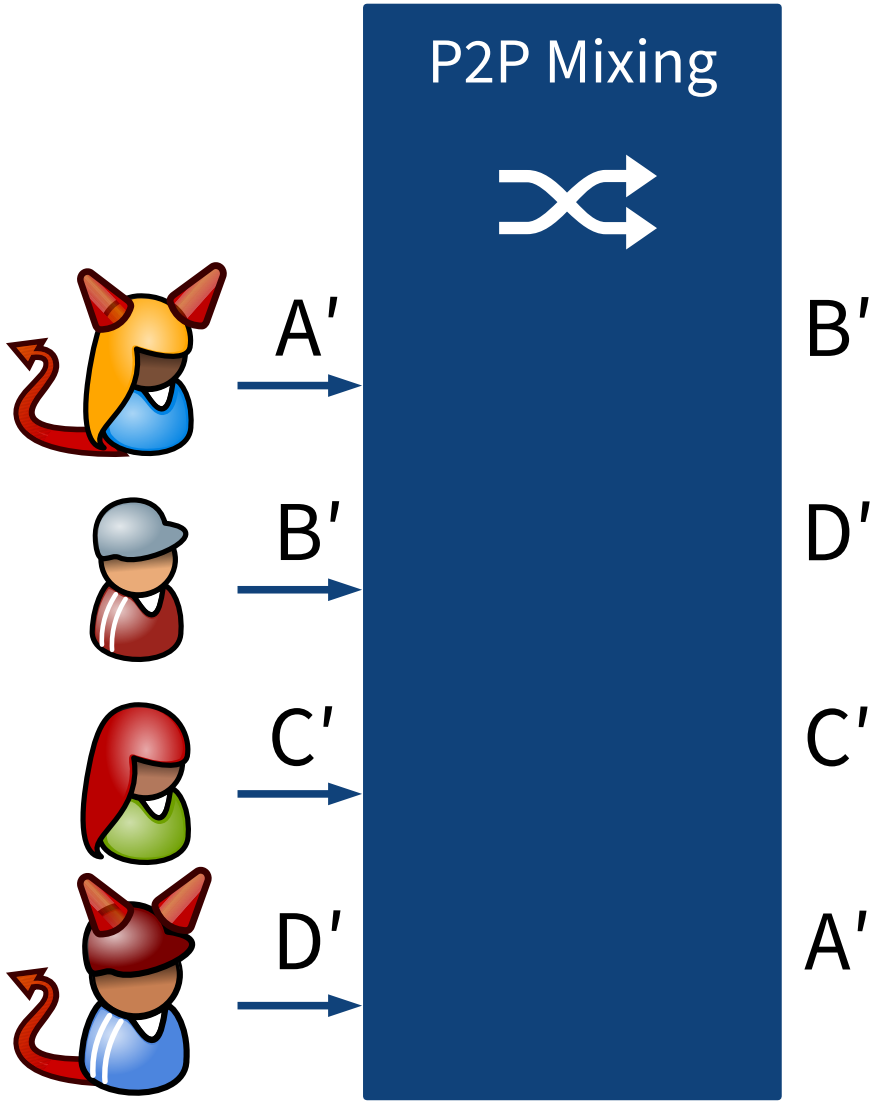
P2P Mixing



P2P Mixing



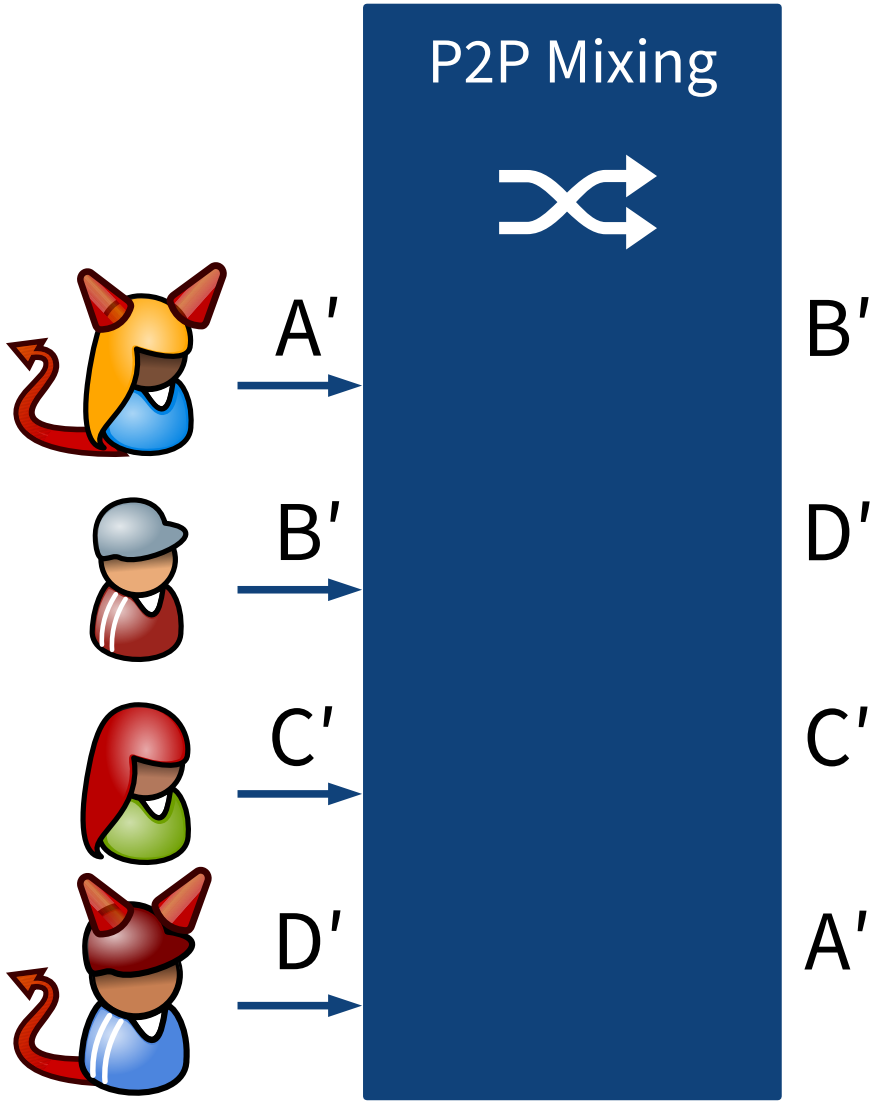
P2P Mixing



P2P Trust Model

No mutual trust,
no third-party routers.

P2P Mixing



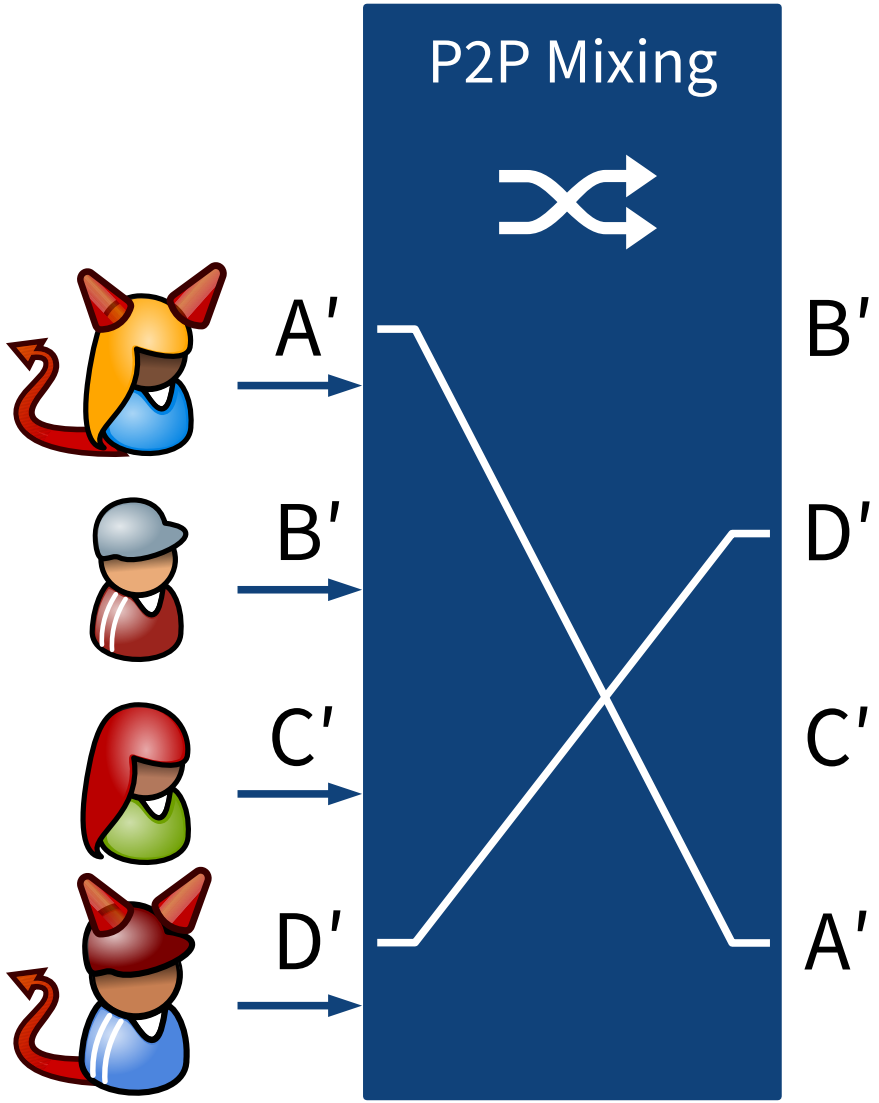
P2P Trust Model

No mutual trust,
no third-party routers.

Anonymity

Anoymity set is the set of
honest users.

P2P Mixing



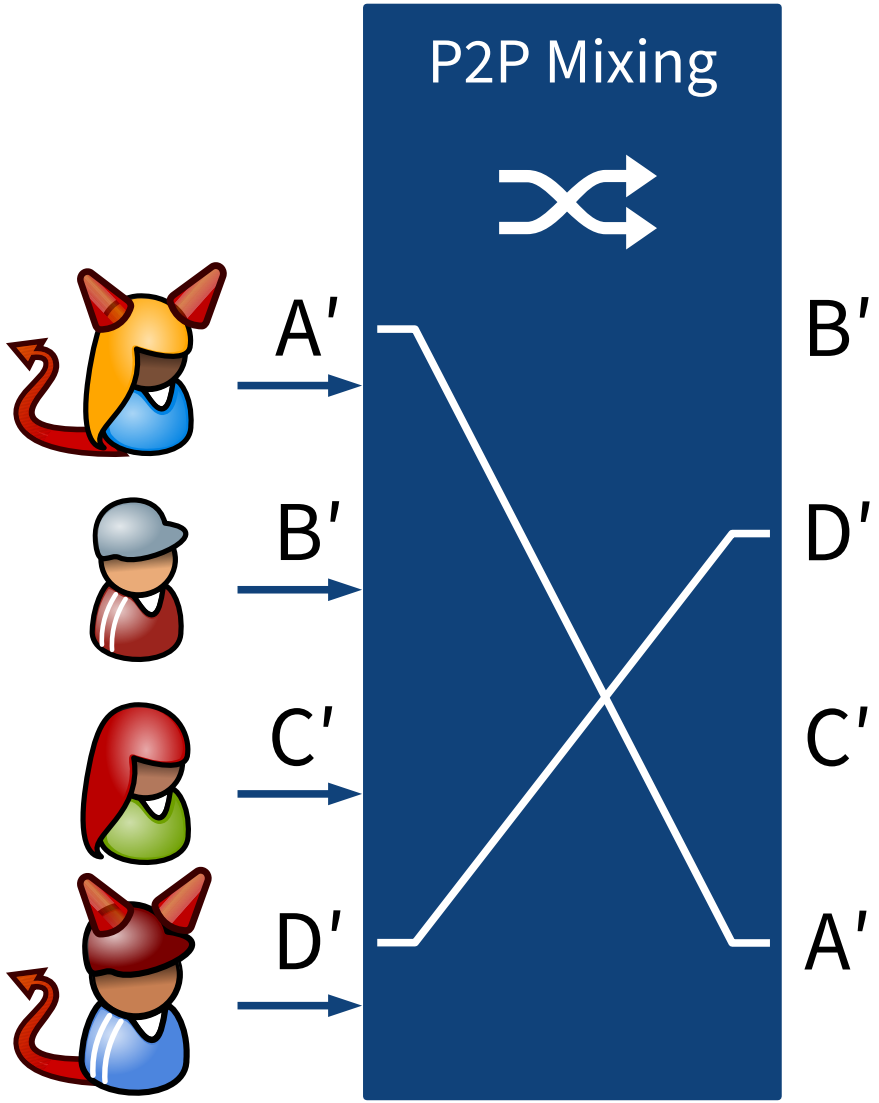
P2P Trust Model

No mutual trust,
no third-party routers.

Anonymity

Anoymity set is the set of
honest users.

P2P Mixing



P2P Trust Model

No mutual trust,
no third-party routers.

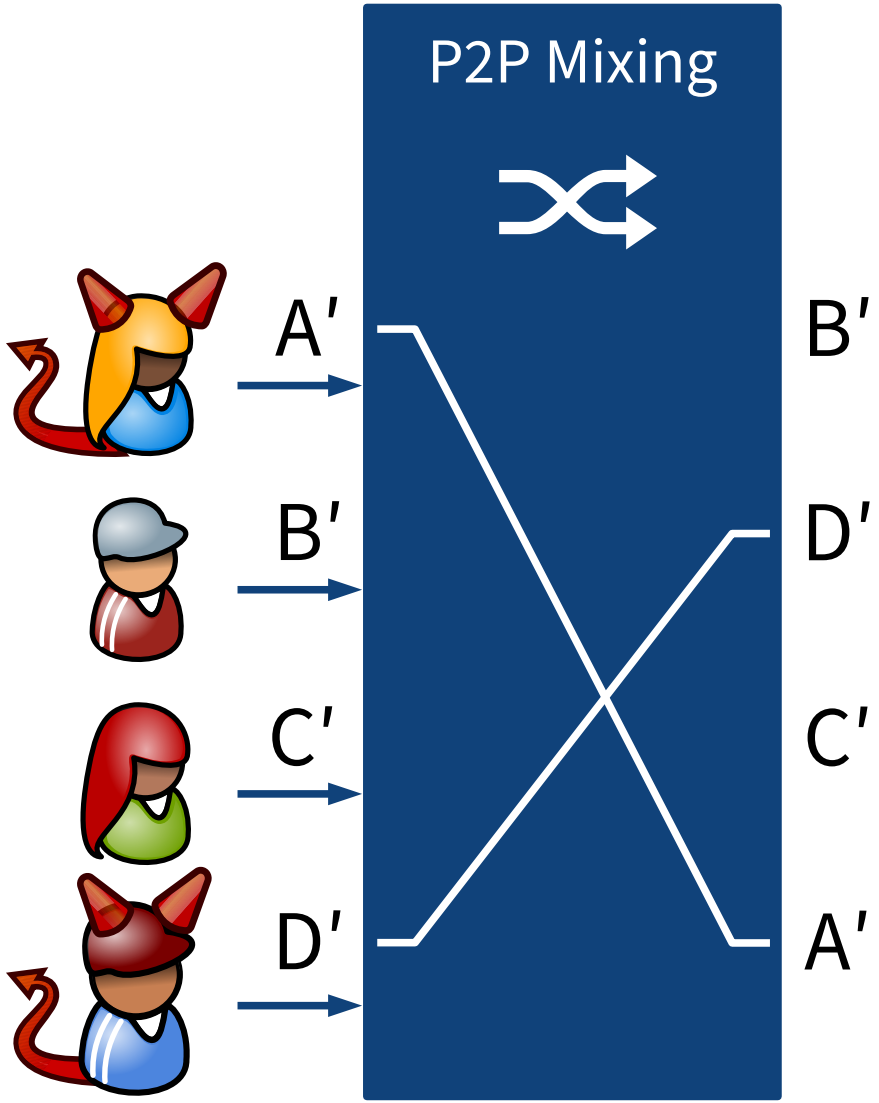
Anonymity

Anoymity set is the set of
honest users.

Termination

Protocol terminates in the
presence of malicious users.

P2P Mixing



P2P Trust Model

No mutual trust,
no third-party routers.

Anonymity

Anoymity set is the set of honest users.

Termination

Protocol terminates in the presence of malicious users.

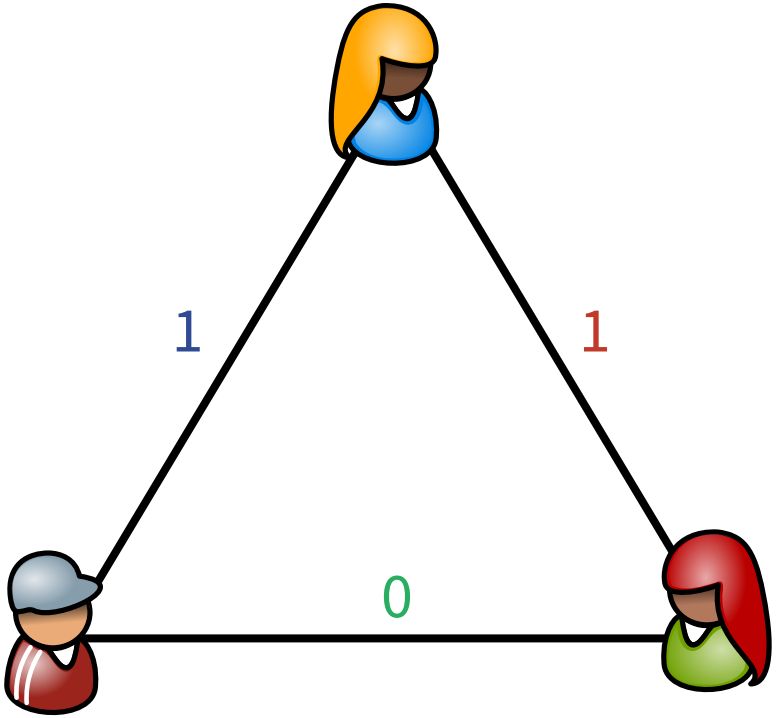
(Bulletin board to facilitate broadcasts.)

CoinShuffle++

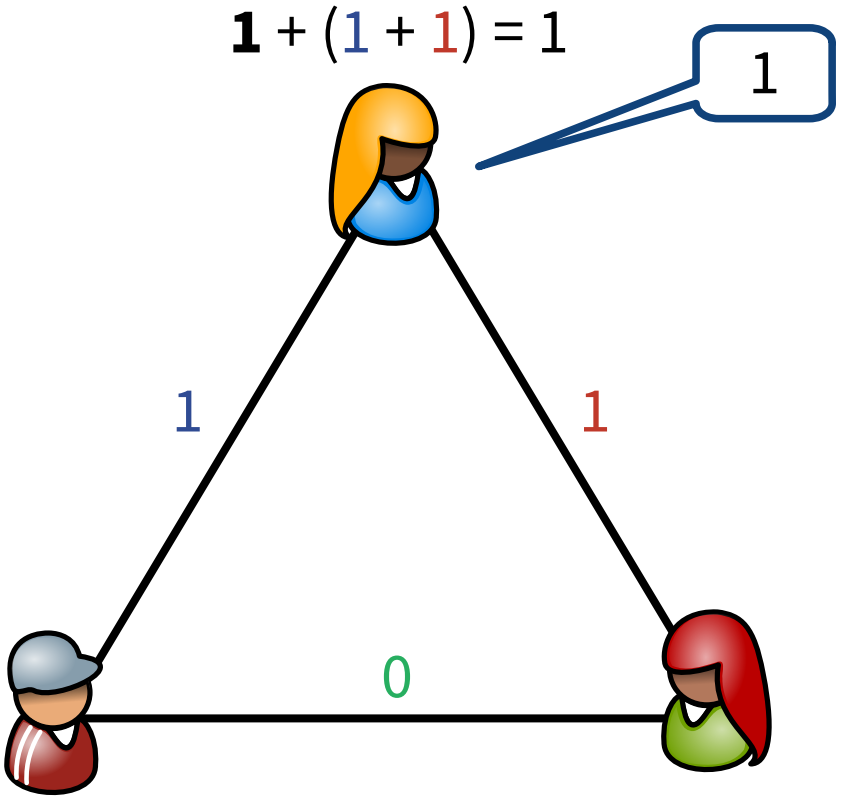
An Efficient P2P Mixing Protocol based on DC-nets

DC-net [Chaum, CRYPTO’88]

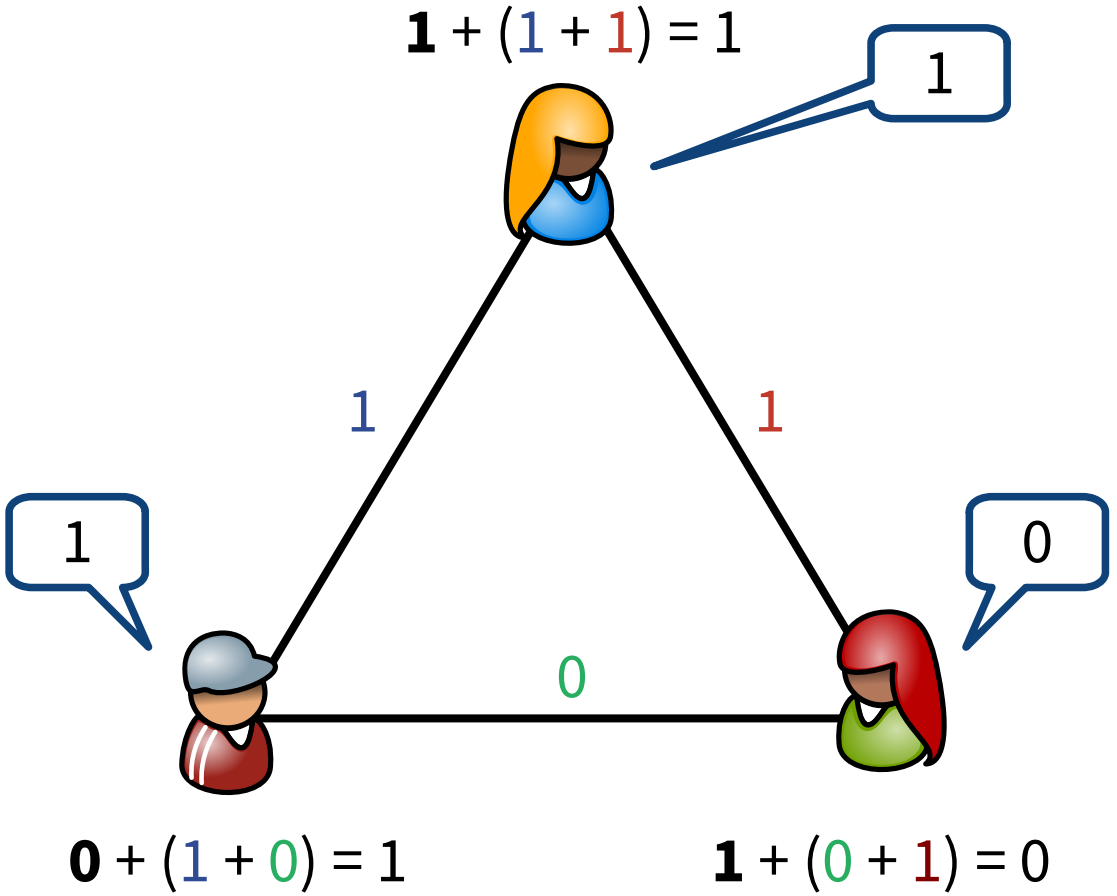
DC-net [Chaum, CRYPTO'88]



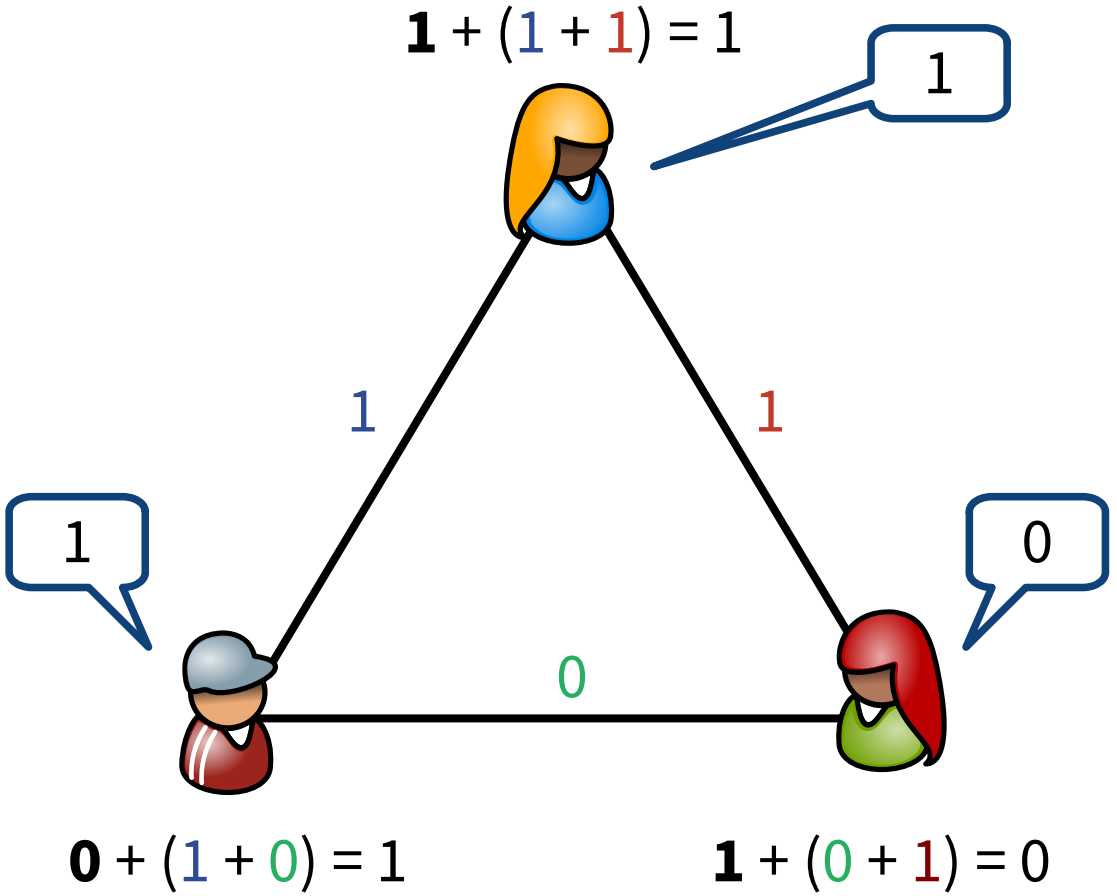
DC-net [Chaum, CRYPTO'88]



DC-net [Chaum, CRYPTO'88]

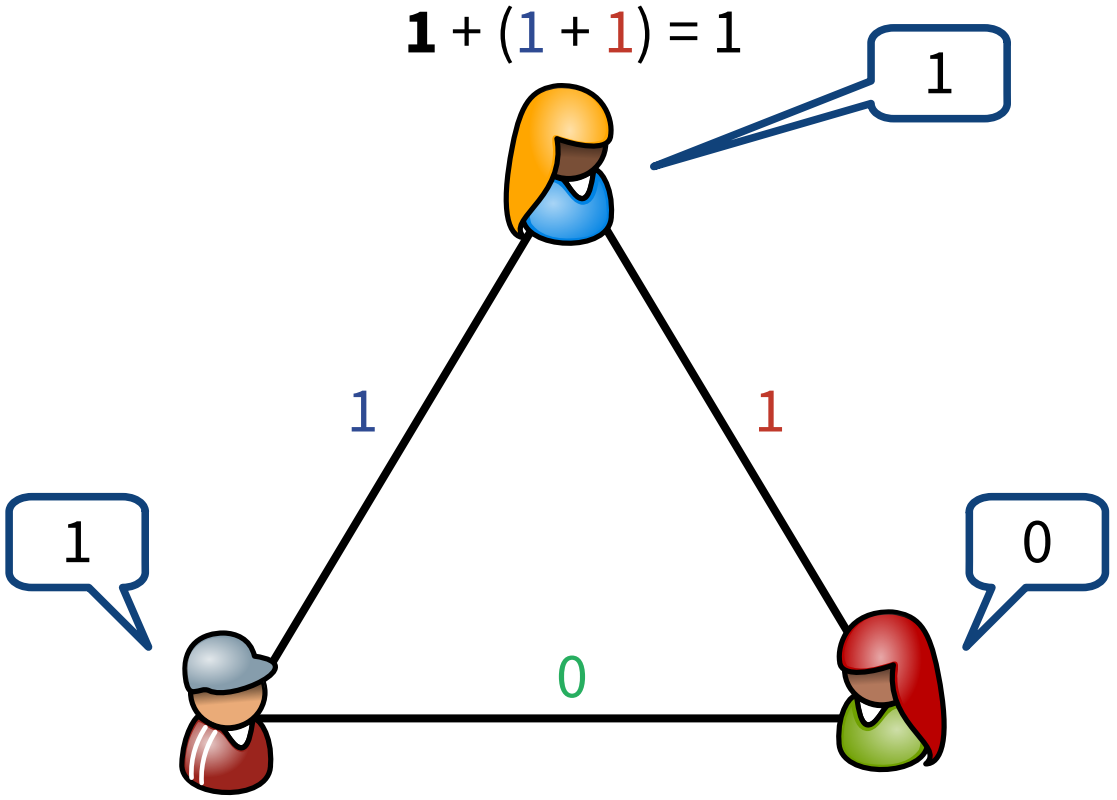


DC-net [Chaum, CRYPTO'88]



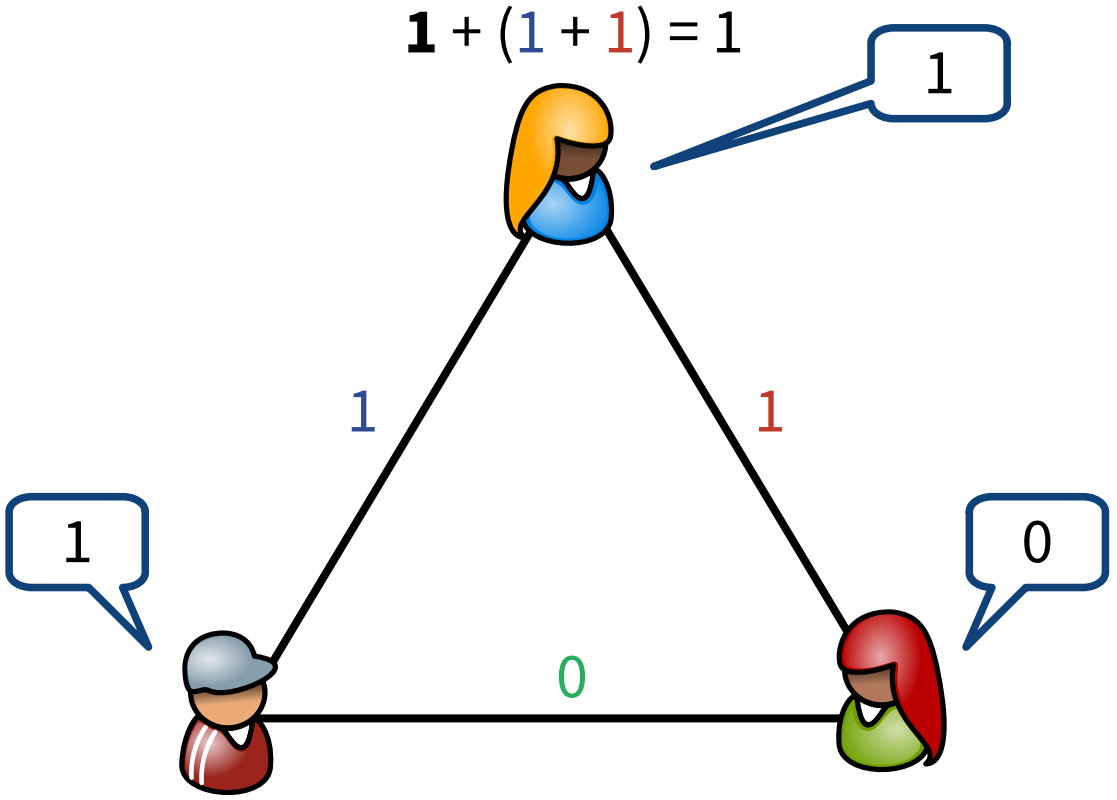
1 + 1 + 0

DC-net [Chaum, CRYPTO'88]



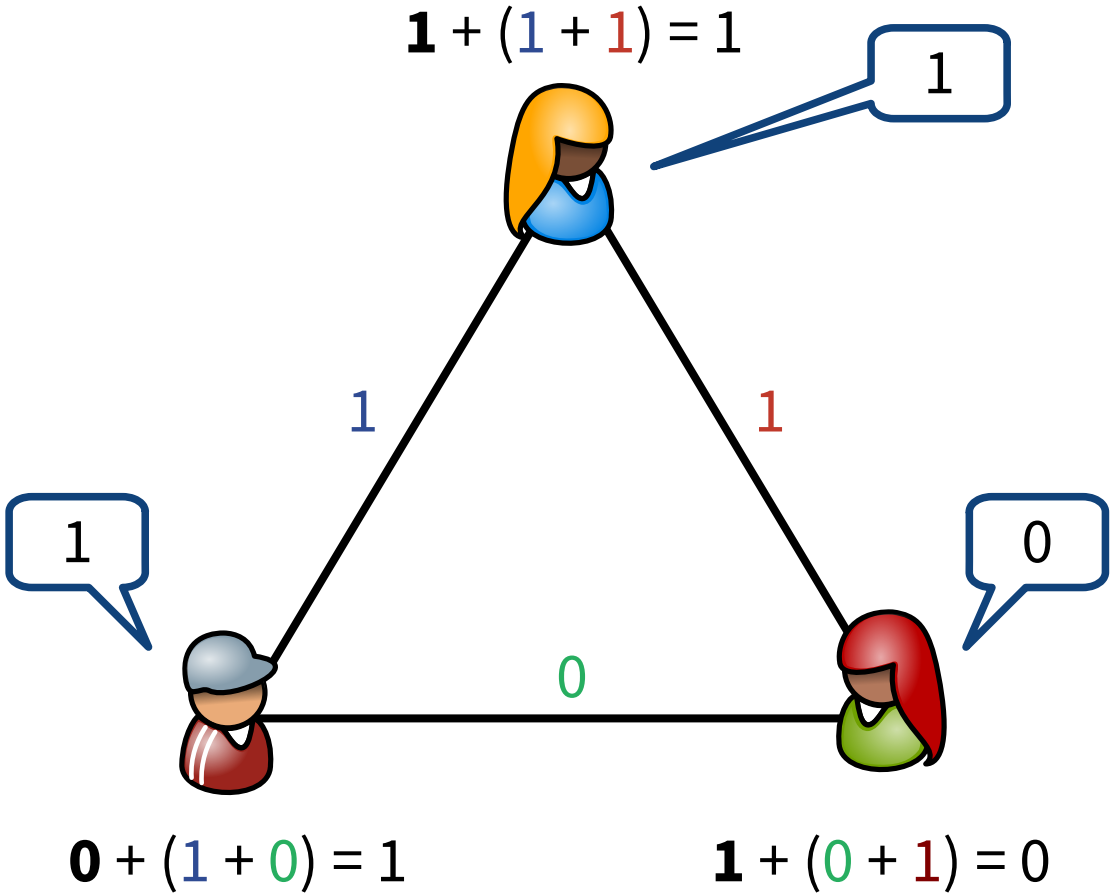
1 + 1 + 0 = [**1** + (1 + 1)] + [**0** + (1 + 0)] + [**1** + (0 + 1)]

DC-net [Chaum, CRYPTO'88]



1 + 1 + 0 = [**1** + (1 + 1)] + [**0** + (1 + 0)] + [**1** + (0 + 1)] = **1** + **1** + 0

DC-net [Chaum, CRYPTO'88]



$$1 + 1 + 0 = [\mathbf{1} + (1 + 1)] + [\mathbf{0} + (1 + 0)] + [\mathbf{1} + (0 + 1)] = \mathbf{1} + \mathbf{1} + 0 = 0$$

DC-nets in Practice

DC-nets in Practice

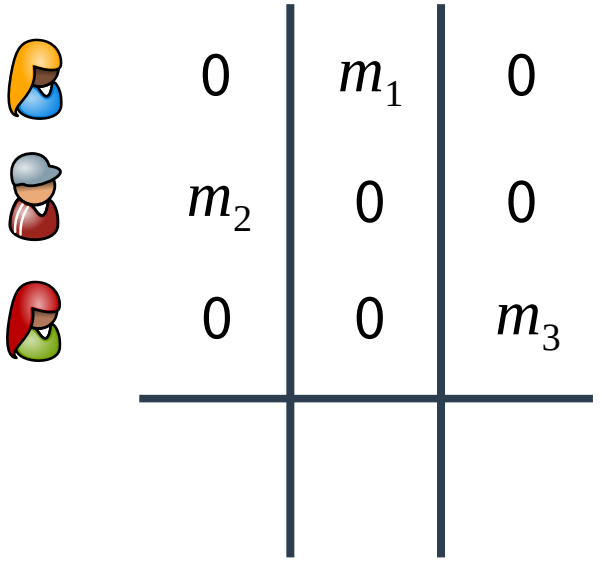
- Obtain shared symmetric keys: perform key exchange

DC-nets in Practice

- Obtain shared symmetric keys: perform key exchange
- Send larger input messages: use larger finite fields

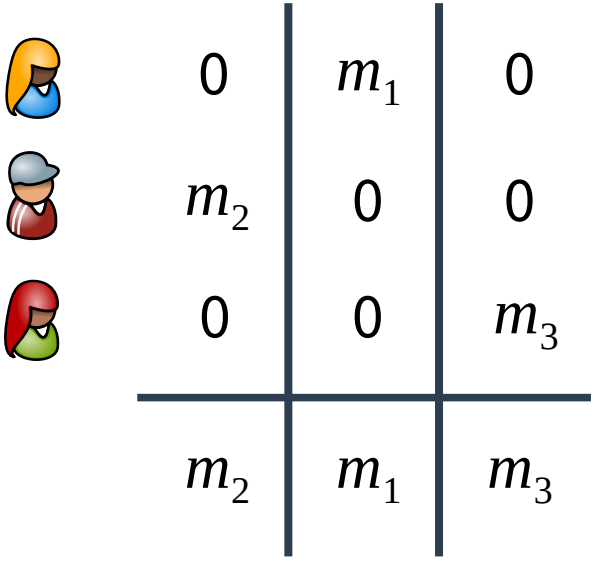
DC-nets in Practice

- Obtain shared symmetric keys: perform key exchange
- Send larger input messages: use larger finite fields
- Compute set of input messages instead of sum: use slots



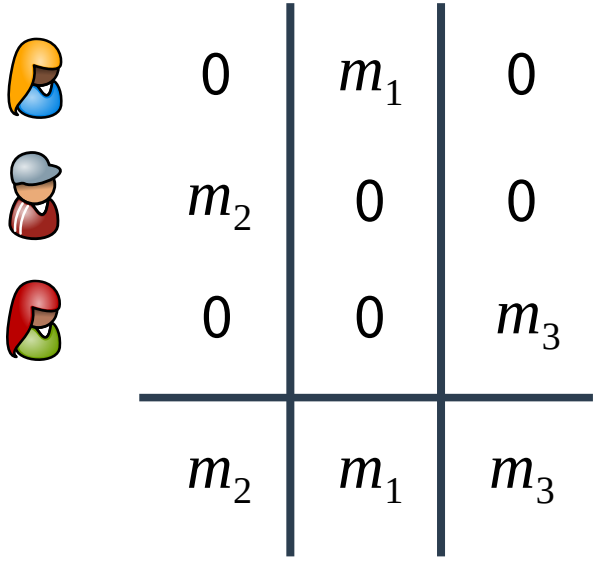
DC-nets in Practice

- Obtain shared symmetric keys: perform key exchange
- Send larger input messages: use larger finite fields
- Compute set of input messages instead of sum: use slots



DC-nets in Practice

- Obtain shared symmetric keys: perform key exchange
- Send larger input messages: use larger finite fields
- Compute set of input messages instead of sum: use slots



Needs anonymous slot assignment!

Multiple Messages (based on [BB, EC'89])


User 1: $m_1 \mid m_1^2 \mid m_1^3 \mid \cdots \mid m_1^n$


User 2:


User 3:


User *n*:

Multiple Messages (based on [BB, EC'89])


User 1:


User 2:


User 3:


User *n*:

m_1	m_1^2	m_1^3	$\cdots$	m_1^n
m_2	m_2^2	m_2^3	$\cdots$	m_2^n
m_3	m_3^2	m_3^3	$\cdots$	m_3^n
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
m_n	m_n^2	m_n^3	$\cdots$	m_n^n

Multiple Messages (based on [BB, EC'89])


User 1:


User 2:


User 3:


User *n*:

m_1	m_1^2	m_1^3	$\cdots$	m_1^n
m_2	m_2^2	m_2^3	$\cdots$	m_2^n
m_3	m_3^2	m_3^3	$\cdots$	m_3^n
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
m_n	m_n^2	m_n^3	$\cdots$	m_n^n
$\sum_{i=1}^{n} m_i$	$\sum_{i=1}^{n} m_i^2$	$\sum_{i=1}^{n} m_i^3$	$\cdots$	$\sum_{i=1}^{n} m_i^n$

Multiple Messages (based on [BB, EC'89])


User 1: m_1 m_1^2 m_1^3 $\cdots$ m_1^n


User 2: m_2 m_2^2 m_2^3 $\cdots$ m_2^n


User 3: m_3 m_3^2 m_3^3 $\cdots$ m_3^n

$\vdots$ $\vdots$ $\vdots$ $\ddots$ $\vdots$


User *n*: m_n m_n^2 m_n^3 $\cdots$ m_n^n

$$\sum_{i=1}^{n} m_i \quad \sum_{i=1}^{n} m_i^2 \quad \sum_{i=1}^{n} m_i^3 \quad \cdots \quad \sum_{i=1}^{n} m_i^n$$

Messages can be recovered from the power sums.

Disruption

User 1:

User 2:

User 3:

User *n*:

$$
\begin{array}{|c|c|c|c|c|}
\hline
m_1 & m_1^2 & m_1^3 & \cdots & m_1^n \\
m_2 & m_2^2 & m_2^3 & \cdots & m_2^n \\
m_3 & m_3^2 & m_3^3 & \cdots & m_3^n \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
m_n & m_n^2 & m_n^3 & \cdots & m_n^n \\
\hline
 & & & \cdots & \\
\hline
\end{array}
$$

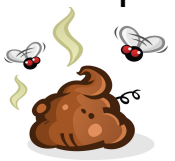
Disruption

User 1:

User 2:

User 3:

User *n*:

m_1	m_1^2	m_1^3	$\cdots$	m_1^n
			$\cdots$	
m_3	m_3^2	m_3^3	$\cdots$	m_3^n
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
m_n	m_n^2	m_n^3	$\cdots$	m_n^n
			$\cdots$	

Disruption

User 1:

User 2:

User 3:

User *n*:

m_1	m_1^2	m_1^3	$\cdots$	m_1^n
m_3	m_3^2	m_3^3	$\cdots$	m_3^n
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
m_n	m_n^2	m_n^3	$\cdots$	m_n^n

Disruption

User 1:

User 2:

User 3:

User *n*:

m_1	m_1^2	m_1^3	$\cdots$	m_1^n
m_3	m_3^2	m_3^3	$\cdots$	m_3^n
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
m_n	m_n^2	m_n^3	$\cdots$	m_n^n

Malicious user stays anonymous!

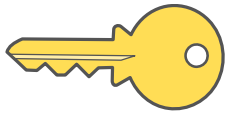
IN CASE OF
DISRUPTION
BREAK ANONYMITY

Flowchart of CoinShuffle++

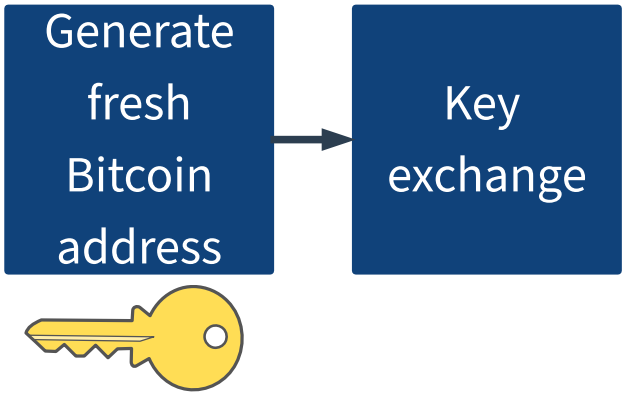
Generate fresh Bitcoin address

Flowchart of CoinShuffle++

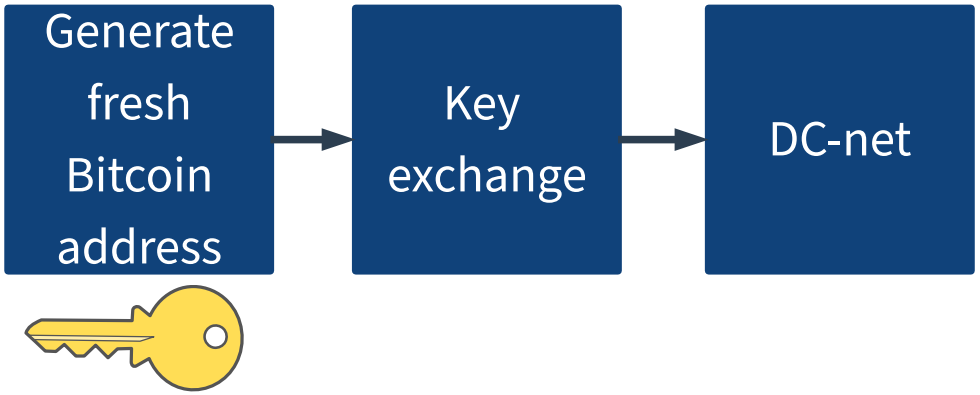
Generate fresh Bitcoin address



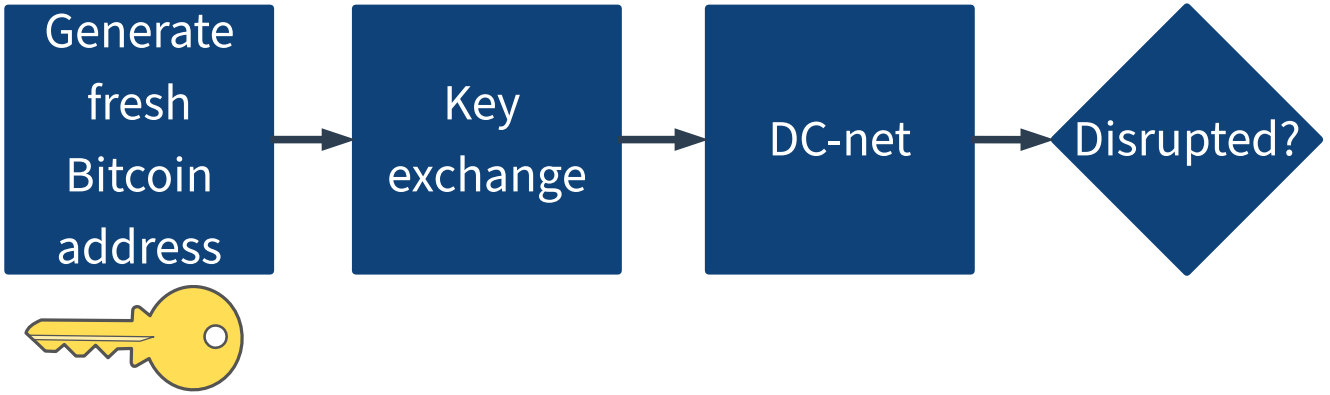
Flowchart of CoinShuffle++



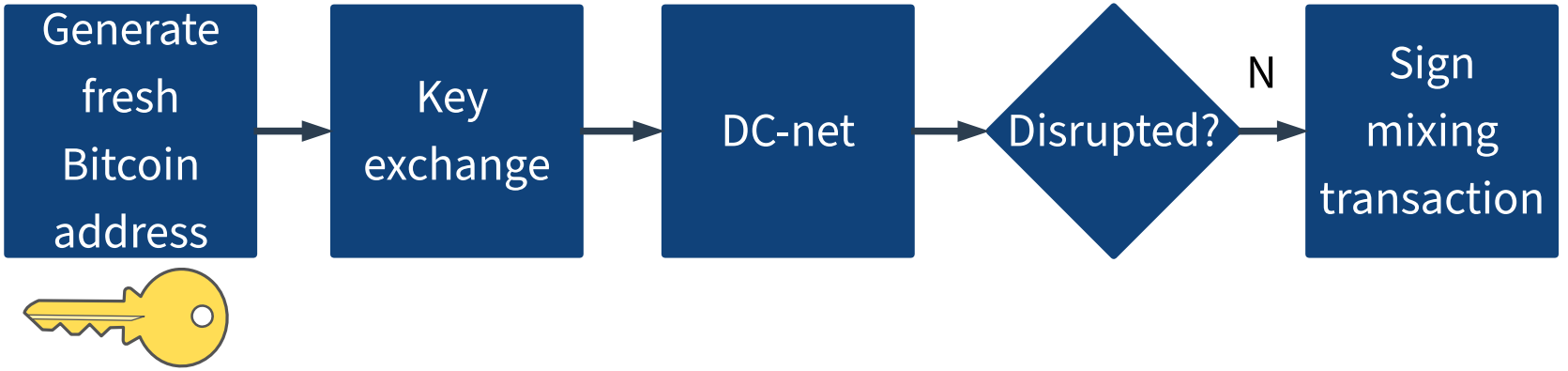
Flowchart of CoinShuffle++



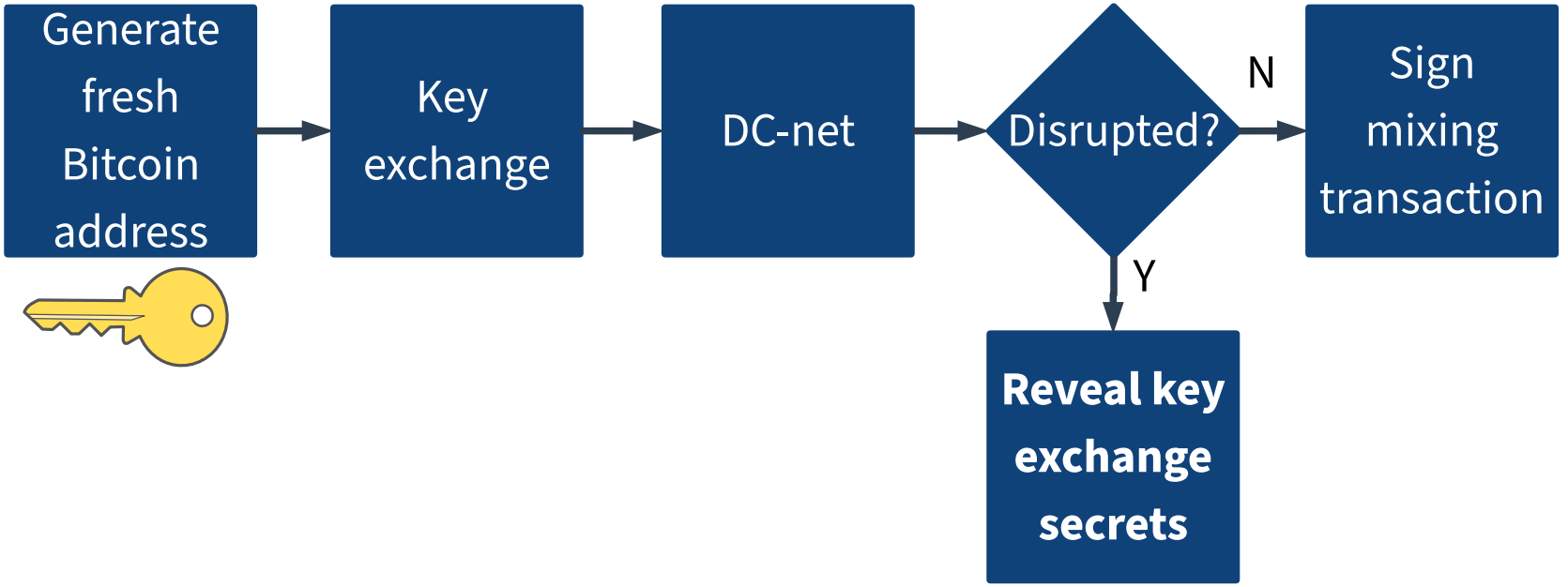
Flowchart of CoinShuffle++



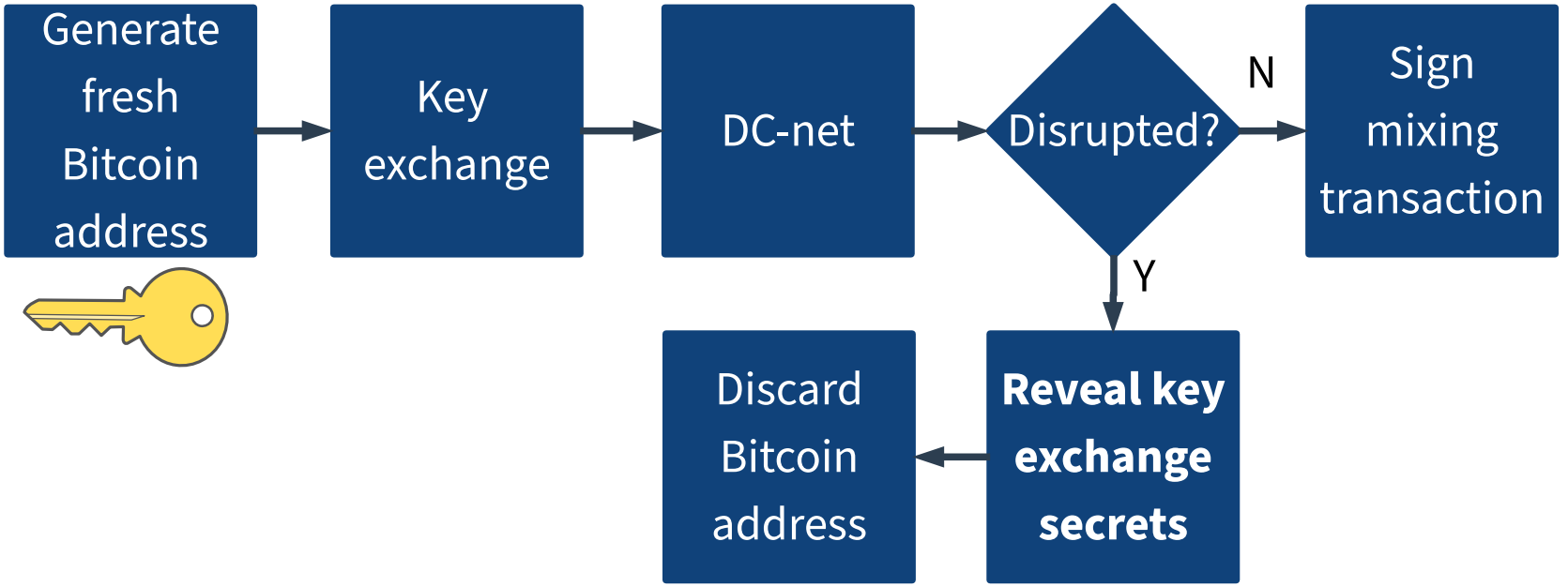
Flowchart of CoinShuffle++

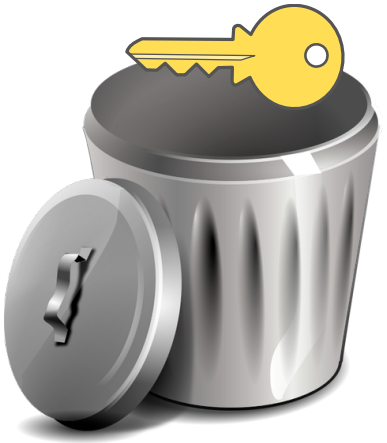


Flowchart of CoinShuffle++

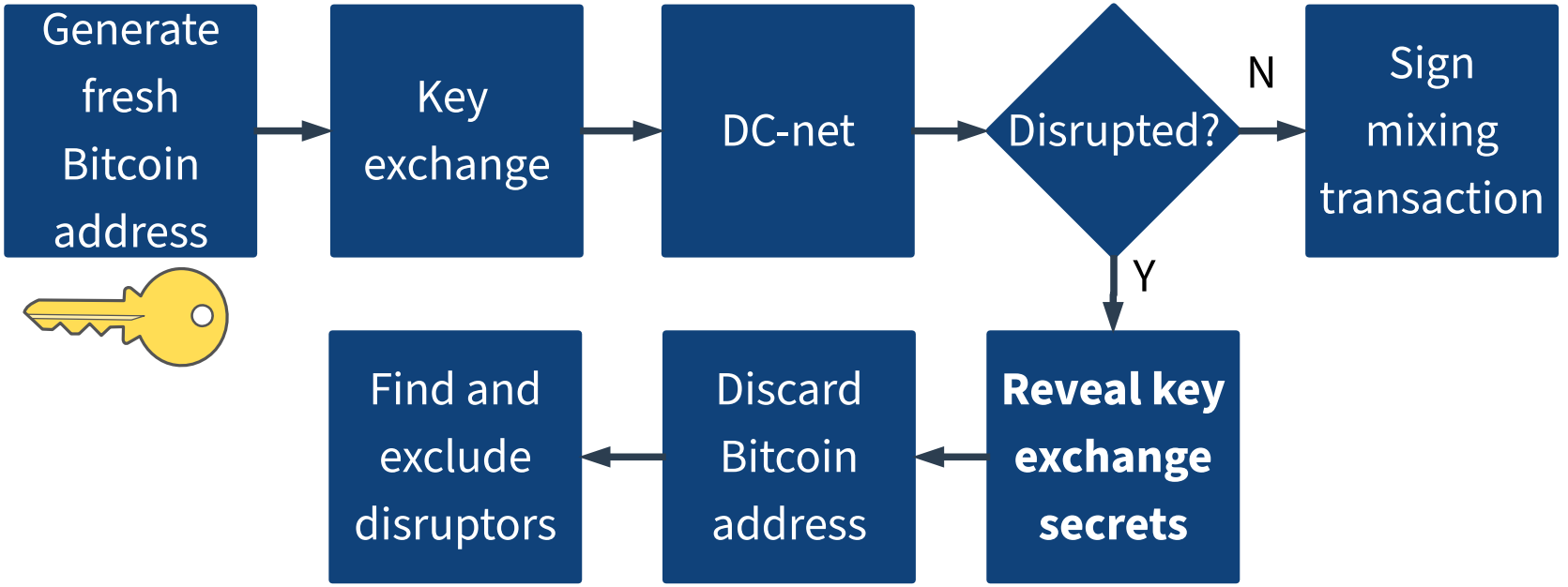


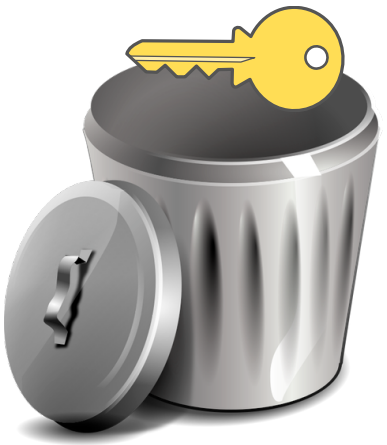
Flowchart of CoinShuffle++



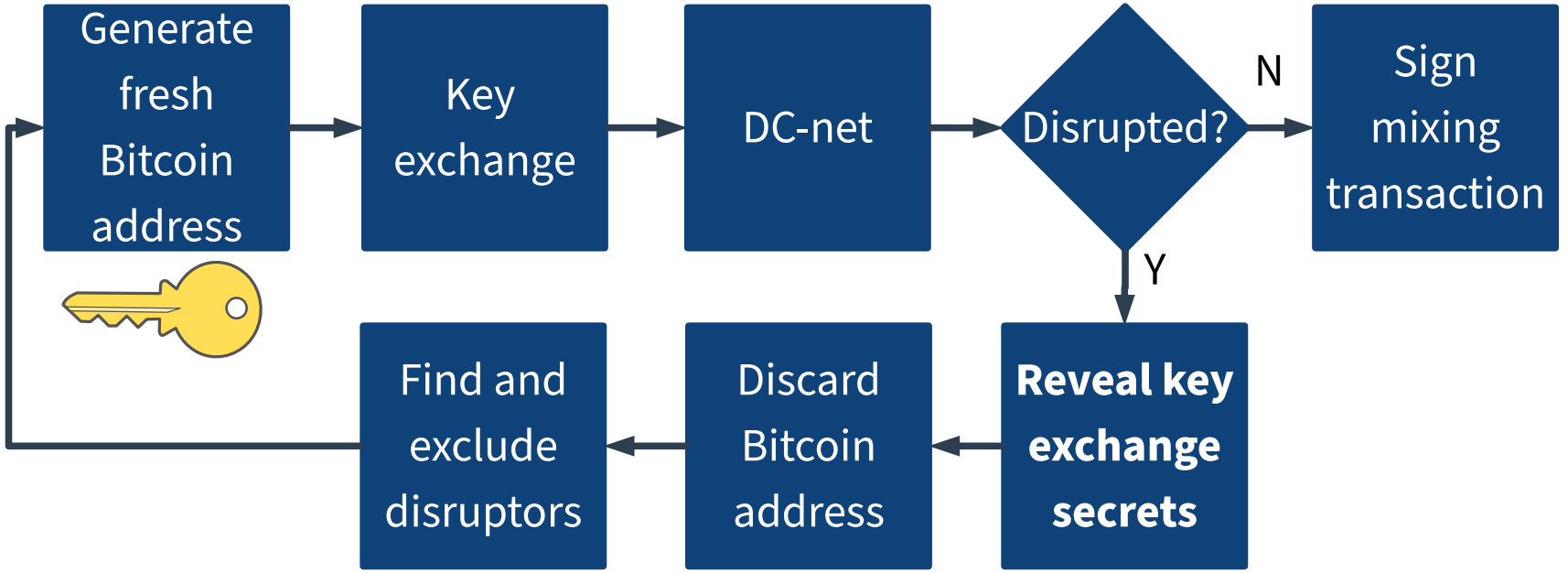


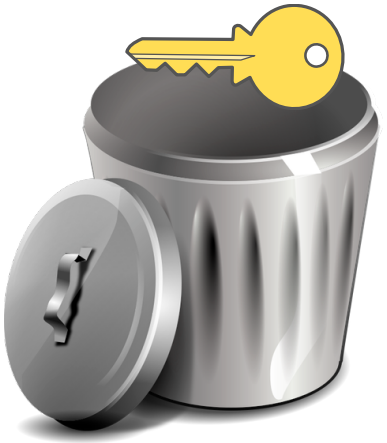
Flowchart of CoinShuffle++



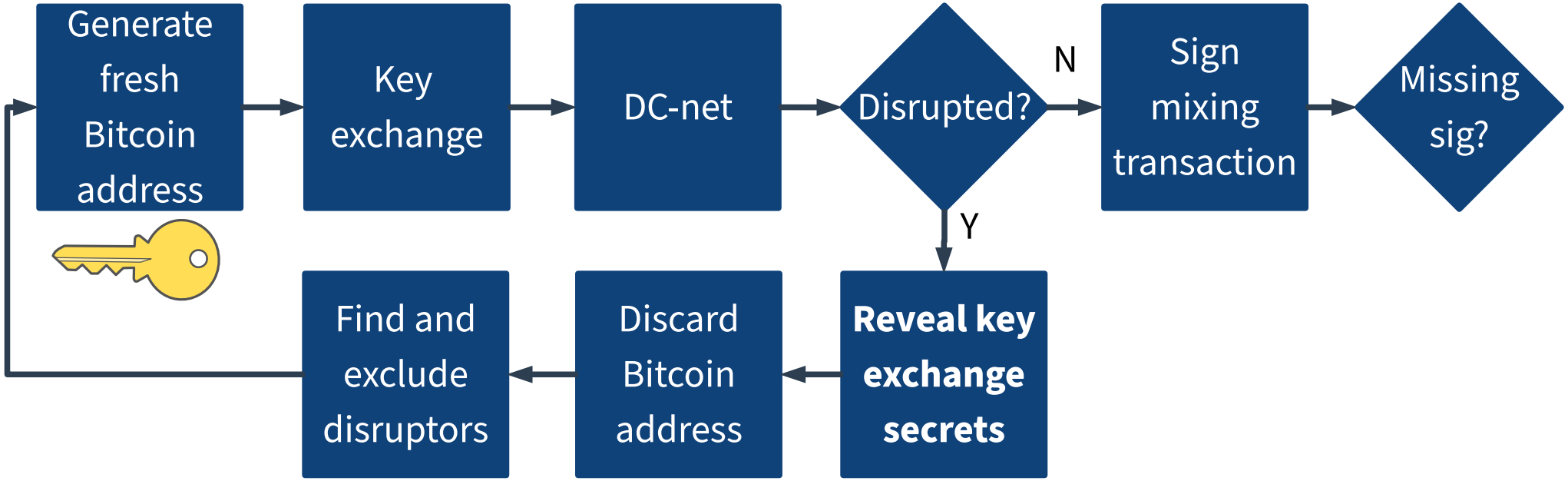


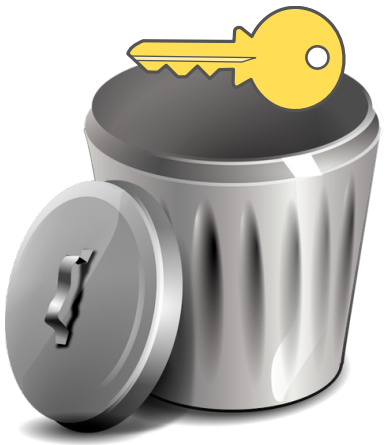
Flowchart of CoinShuffle++



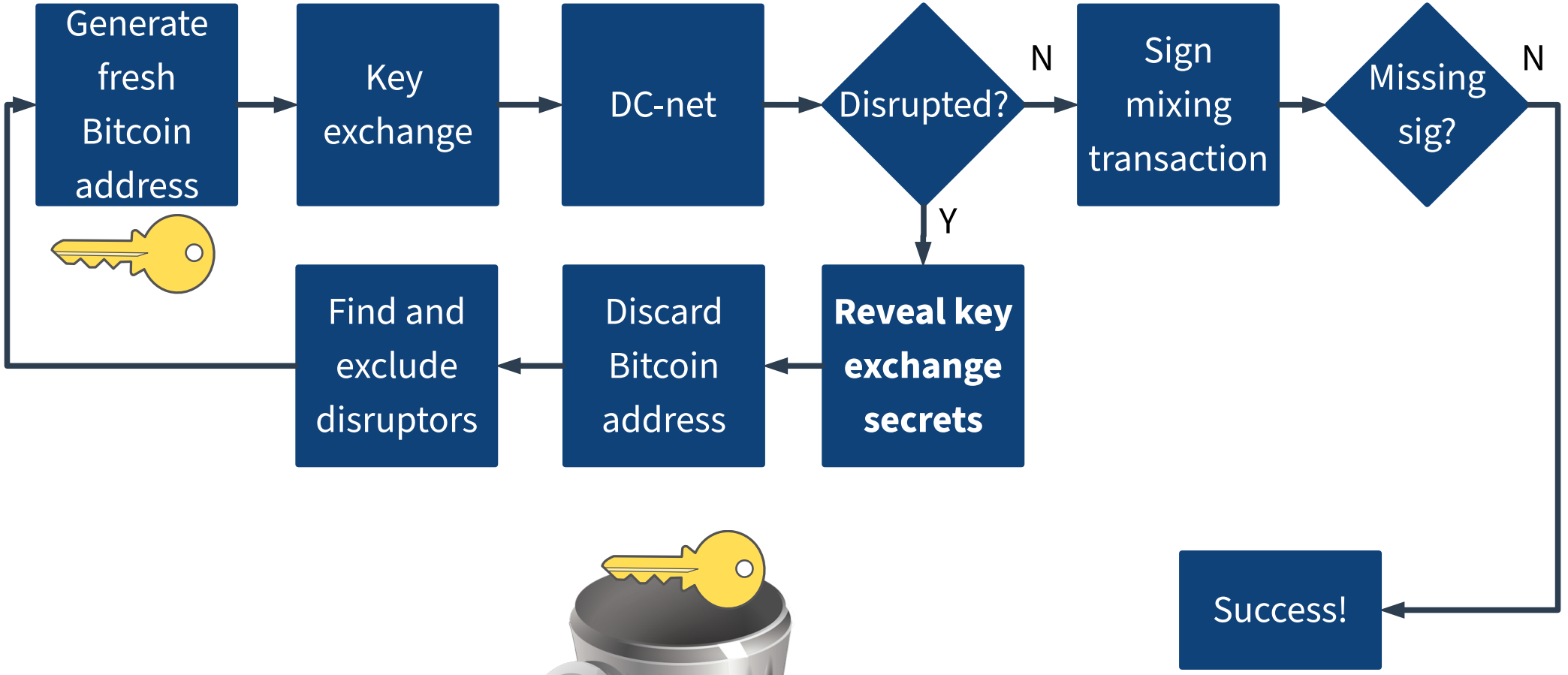


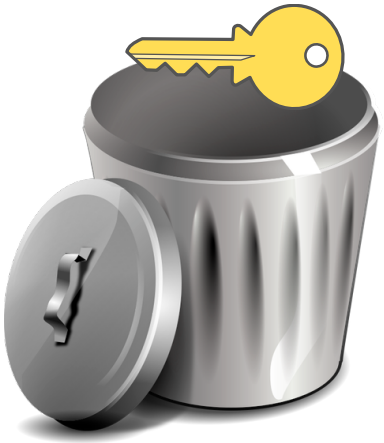
Flowchart of CoinShuffle++



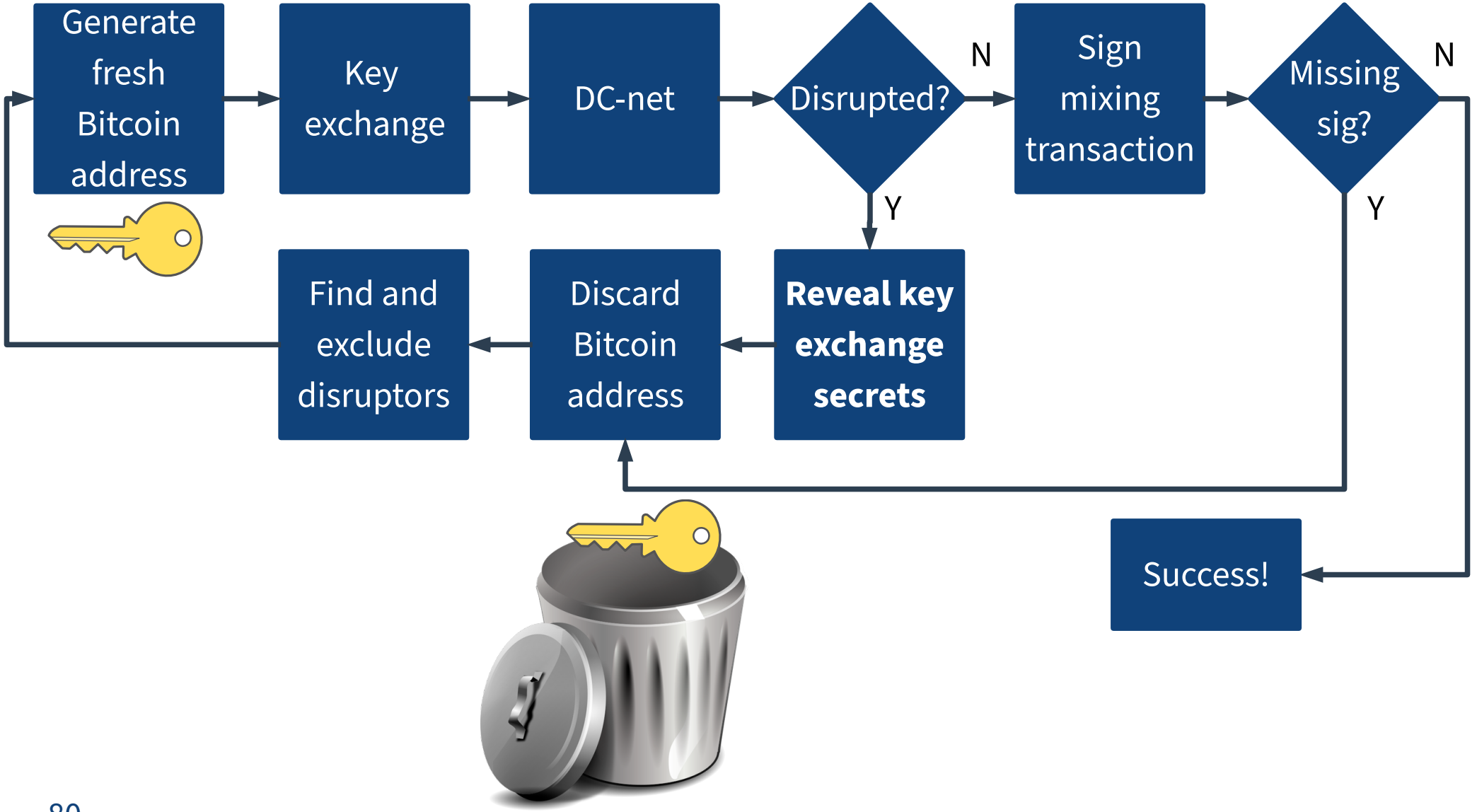


Flowchart of CoinShuffle++

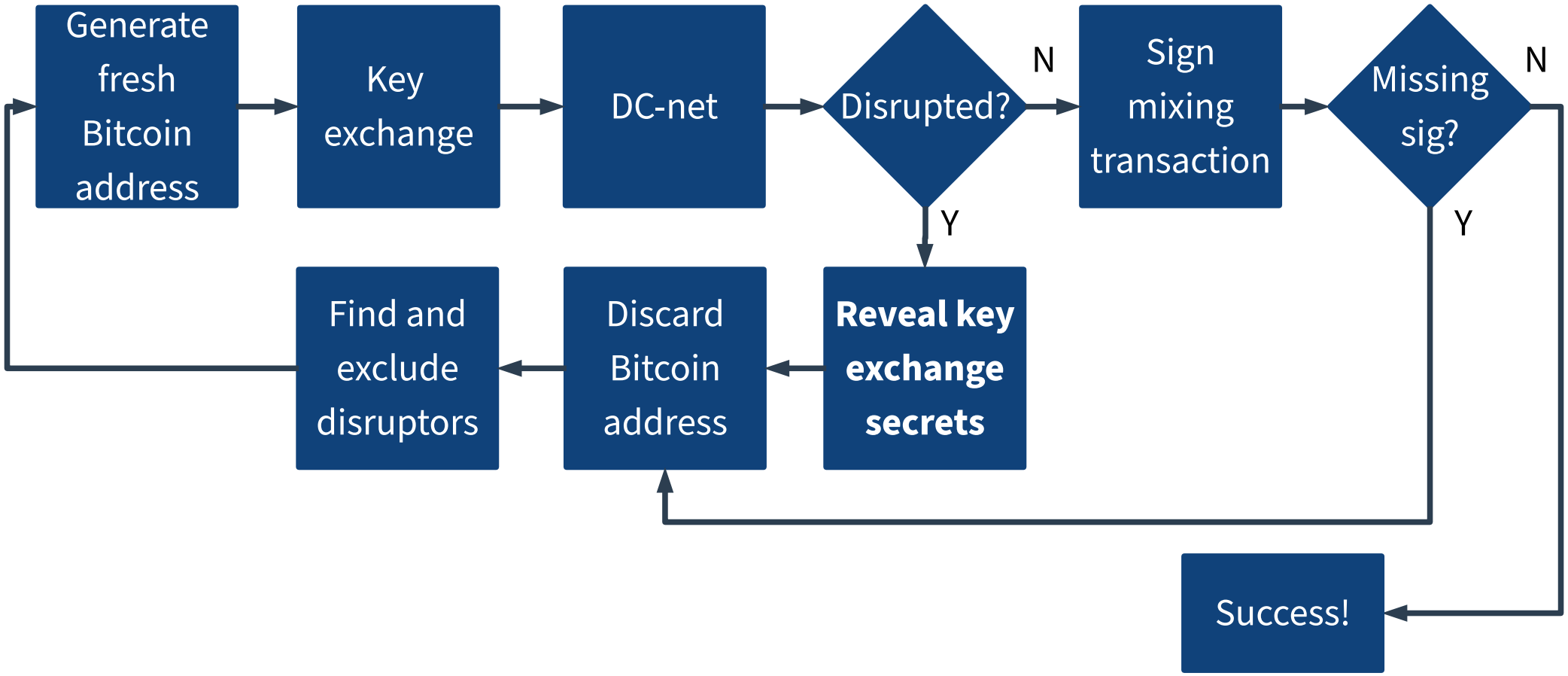




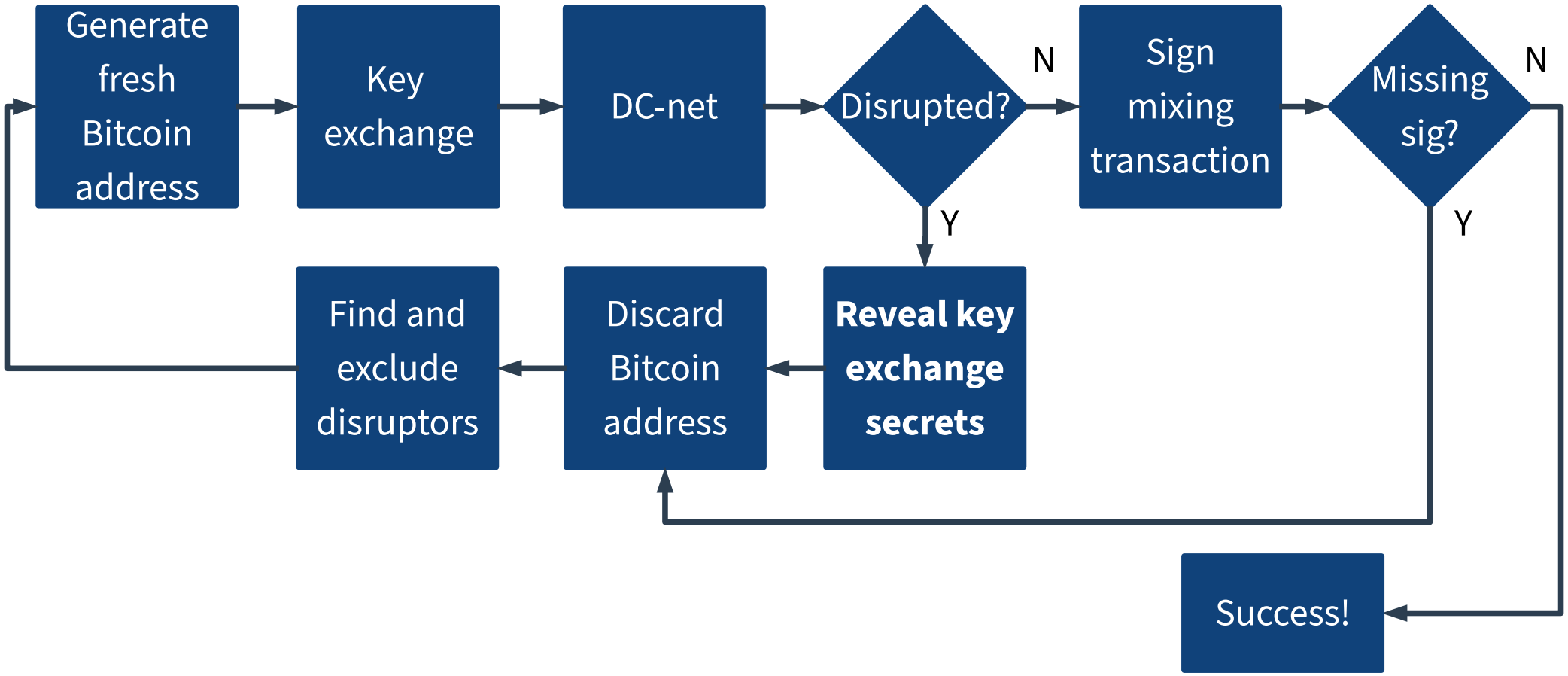
Flowchart of CoinShuffle++



Flowchart of CoinShuffle++



Flowchart of CoinShuffle++

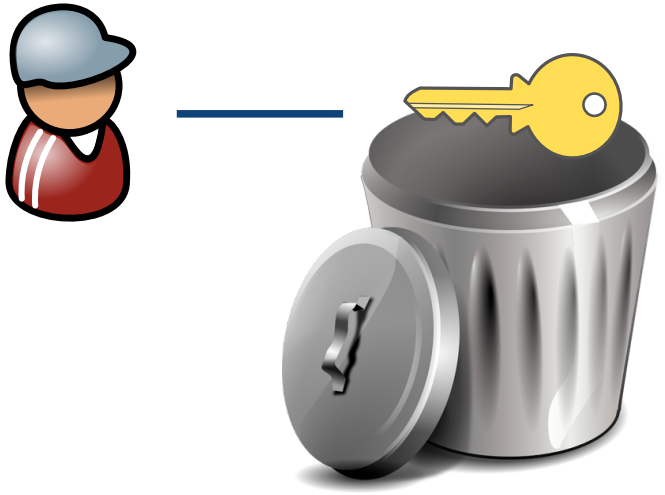


CoinShuffle++ provides anonymity and termination.

Side Topic: Are Fresh Messages Needed?

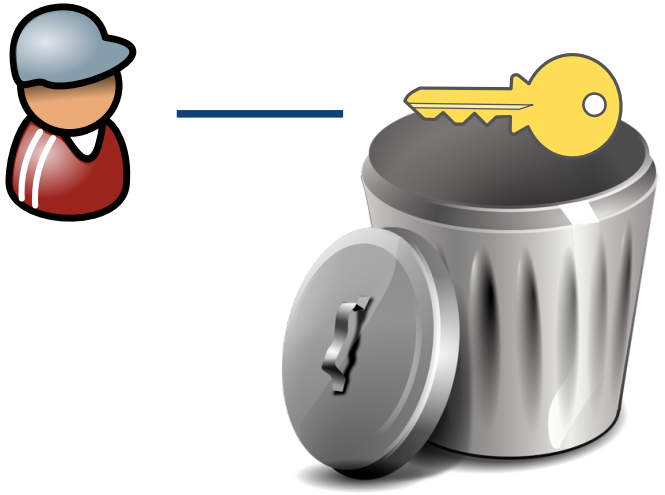
“Fresh” vs. “Fixed” Input Messages

Fresh:



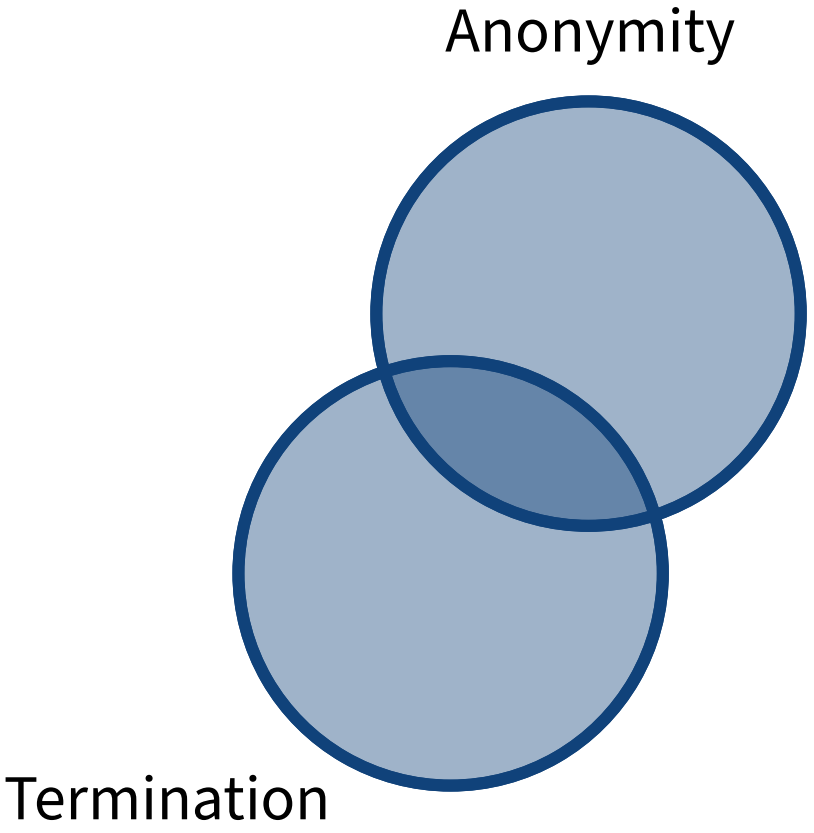
“Fresh” vs. “Fixed” Input Messages

Fresh:

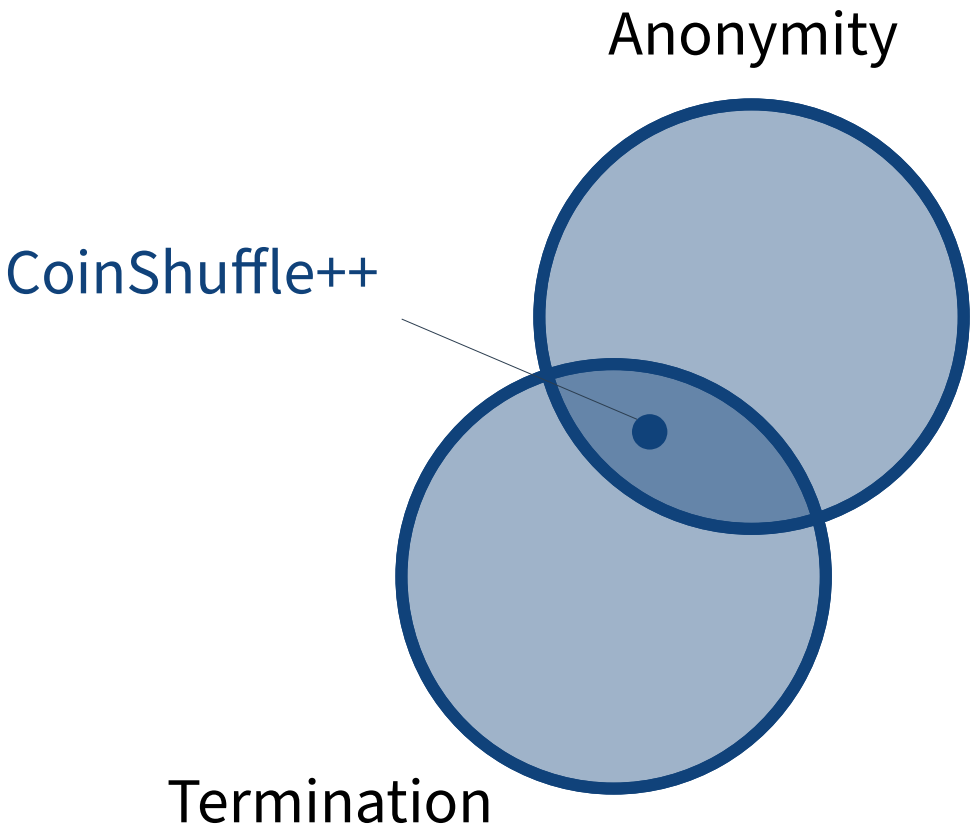


Fixed:

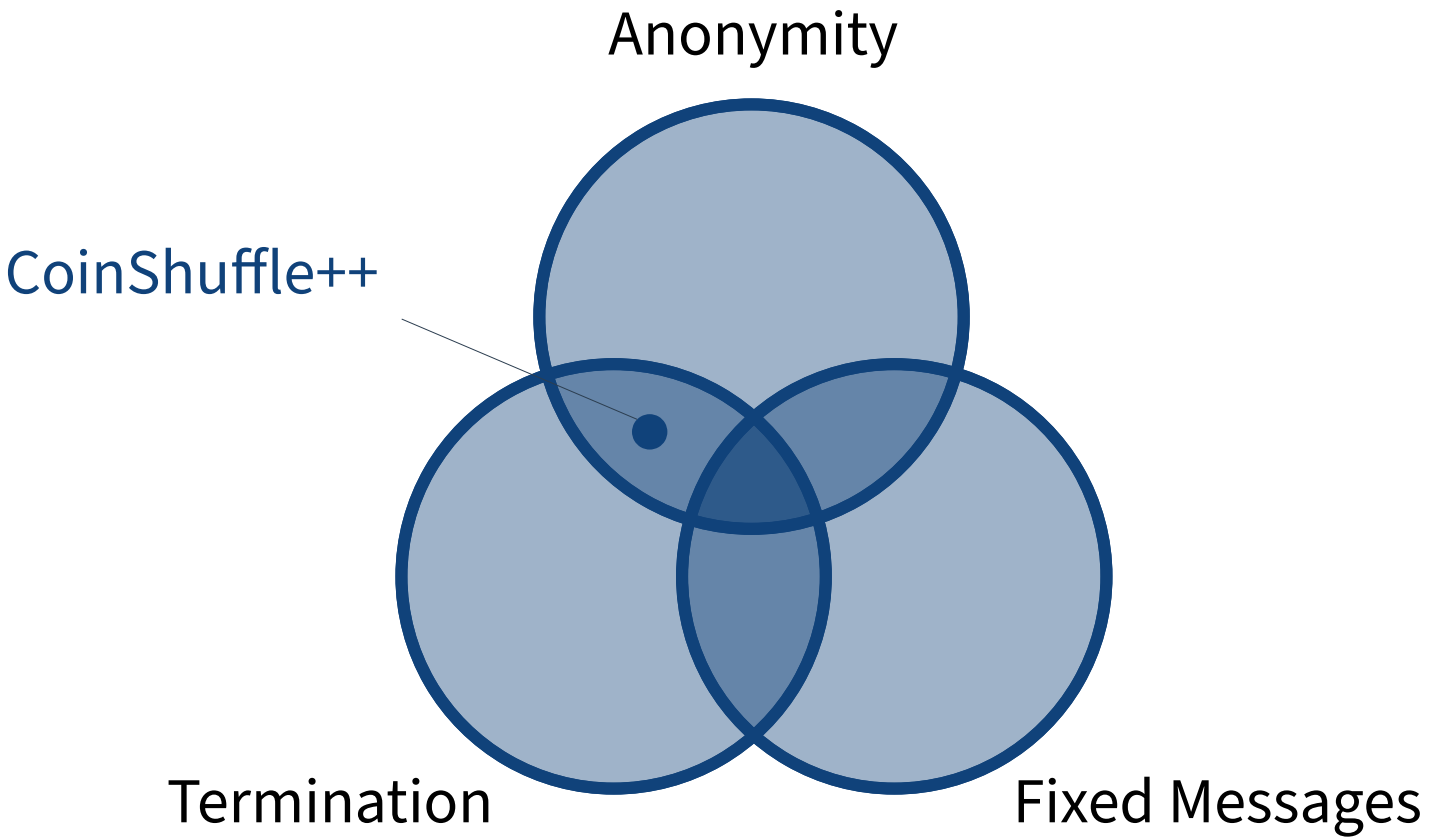
Features of P2P Mixing Protocols



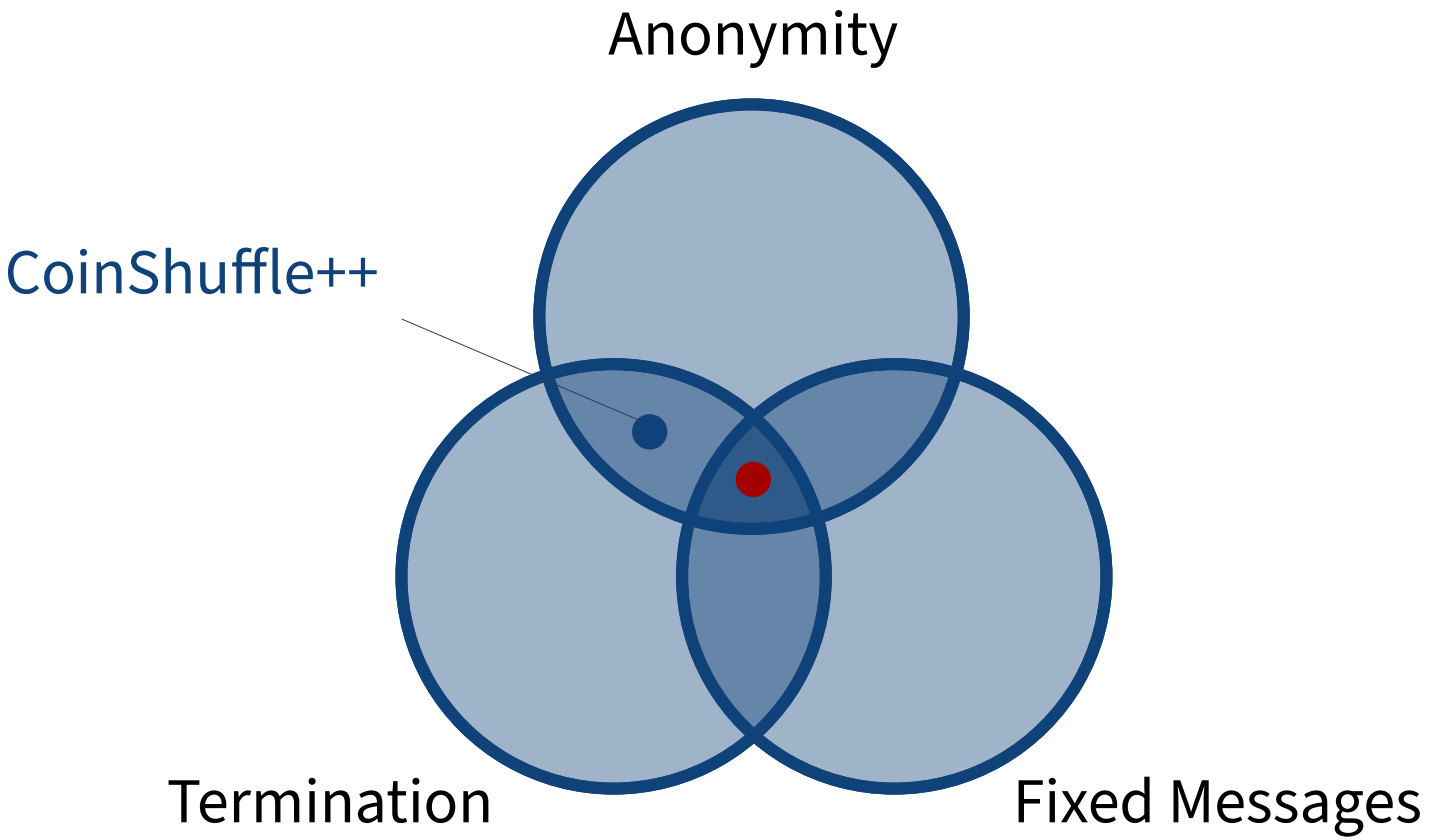
Features of P2P Mixing Protocols



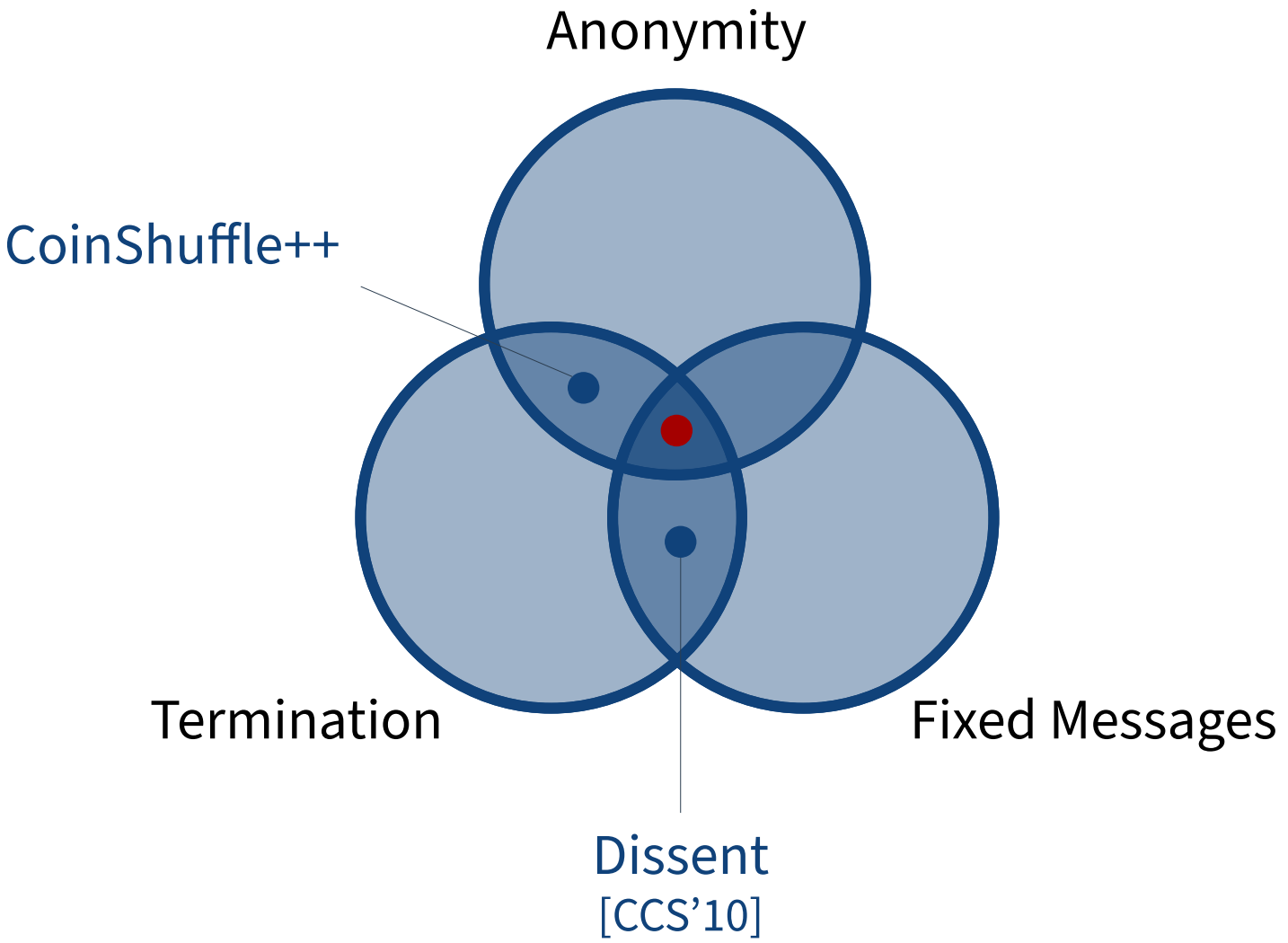
Features of P2P Mixing Protocols



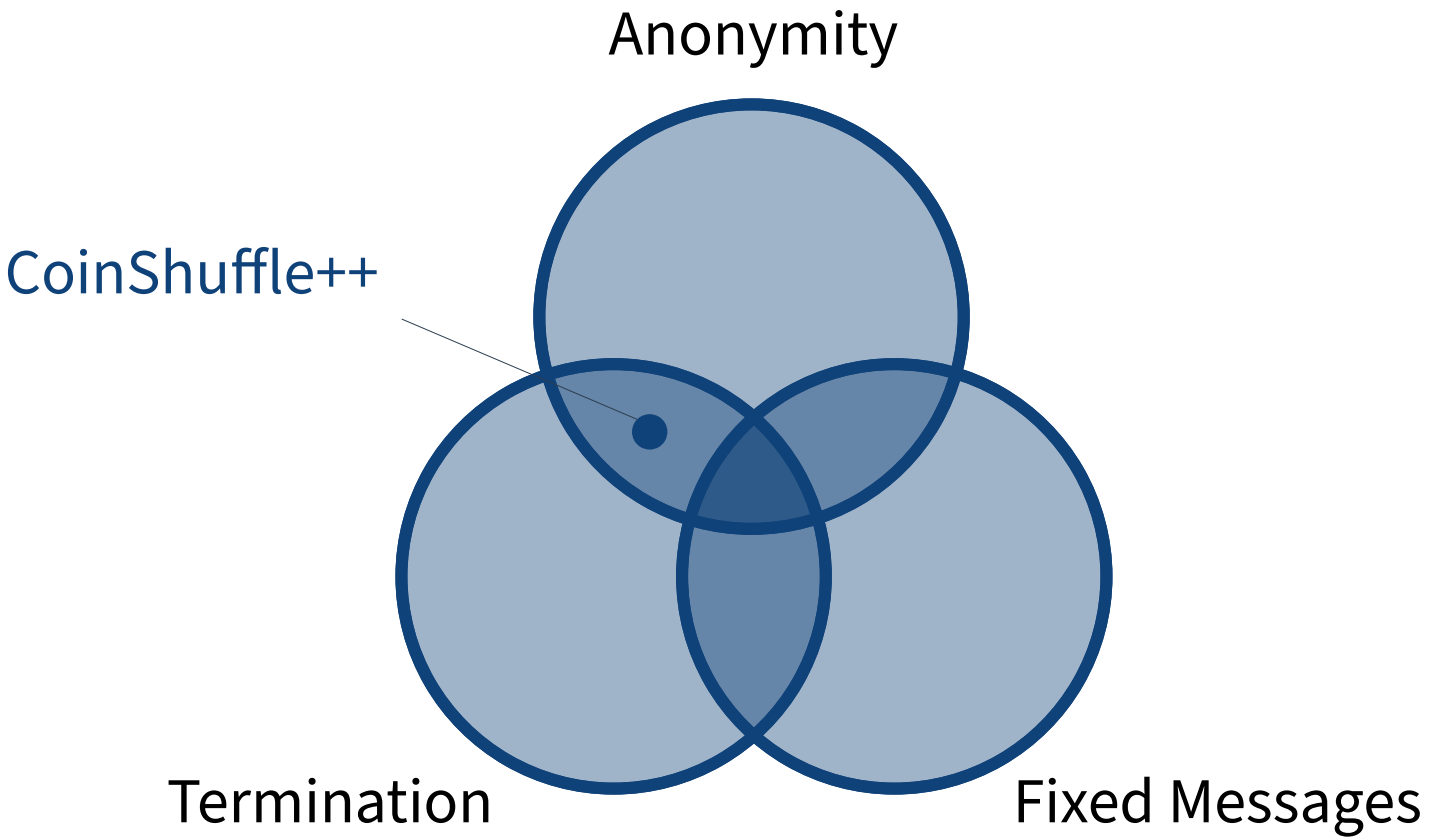
Features of P2P Mixing Protocols



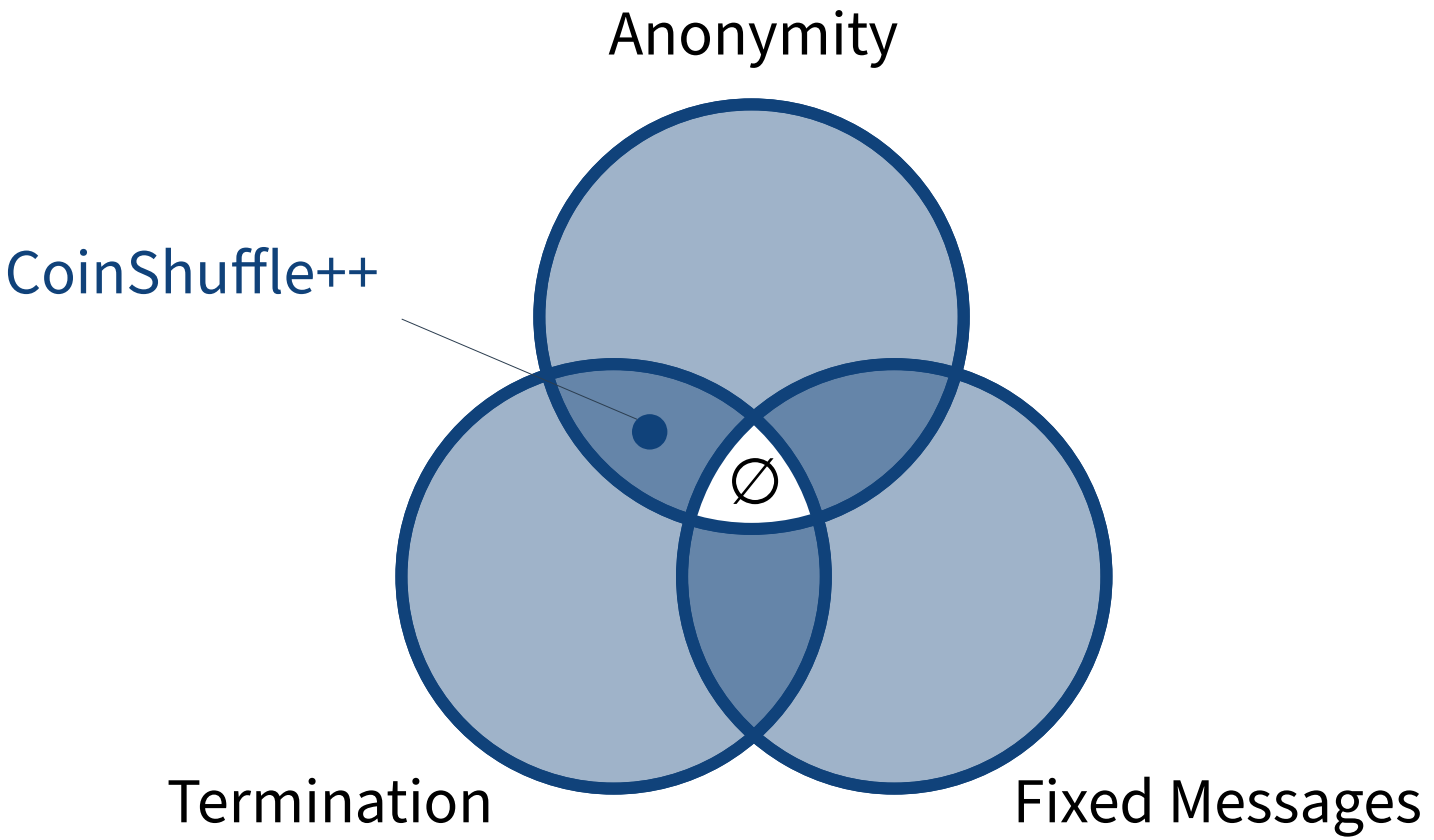
Features of P2P Mixing Protocols



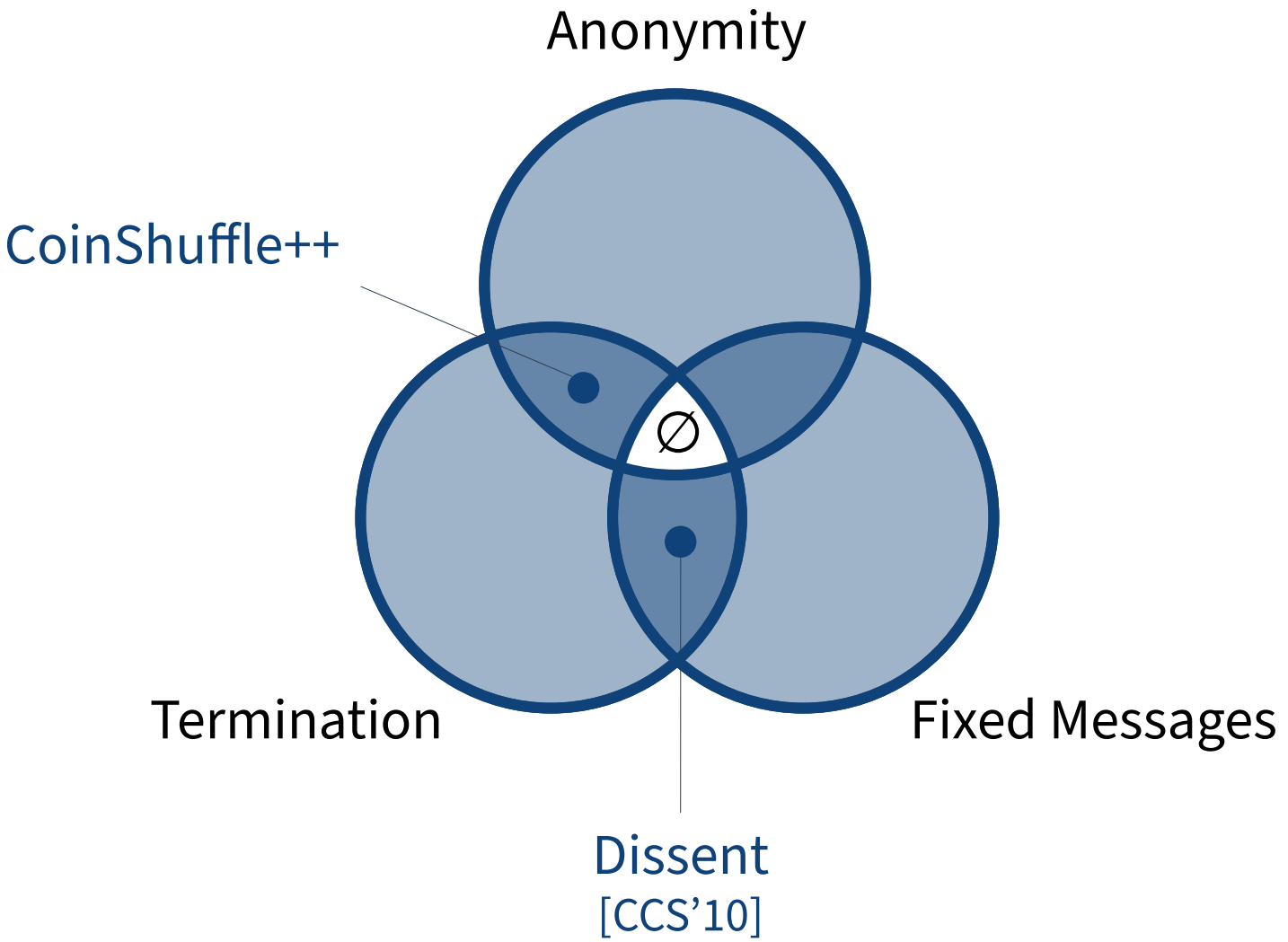
P2P Mixing Trilemma



P2P Mixing Trilemma



P2P Mixing Trilemma



Attack on “Dissent” Protocol

Attack on “Dissent” Protocol

- Dissent proceeds in broadcast rounds

Attack on “Dissent” Protocol

- Dissent proceeds in broadcast rounds
- Outcome of the protocol is revealed to users in last broadcast

Attack on “Dissent” Protocol

- Dissent proceeds in broadcast rounds
- Outcome of the protocol is revealed to users in last broadcast
- Also network attacker sees the outcome in the last broadcast

Attack on “Dissent” Protocol

- Dissent proceeds in broadcast rounds
- Outcome of the protocol is revealed to users in last broadcast
- Also network attacker sees the outcome in the last broadcast

Attack on “Dissent” Protocol

- Dissent proceeds in broadcast rounds
- Outcome of the protocol is revealed to users in last broadcast
- Also network attacker sees the outcome in the last broadcast

Attack on “Dissent” Protocol

- Dissent proceeds in broadcast rounds
- Outcome of the protocol is revealed to users in last broadcast
- Also network attacker sees the outcome in the last broadcast

Run 1:

Attack on “Dissent” Protocol

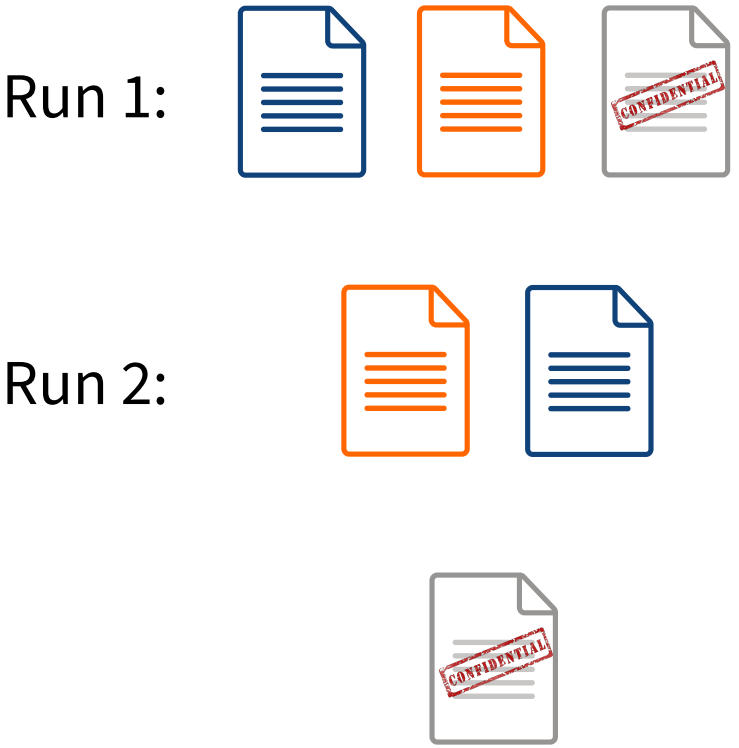
- Dissent proceeds in broadcast rounds
- Outcome of the protocol is revealed to users in last broadcast
- Also network attacker sees the outcome in the last broadcast

Run 1:

Run 2:

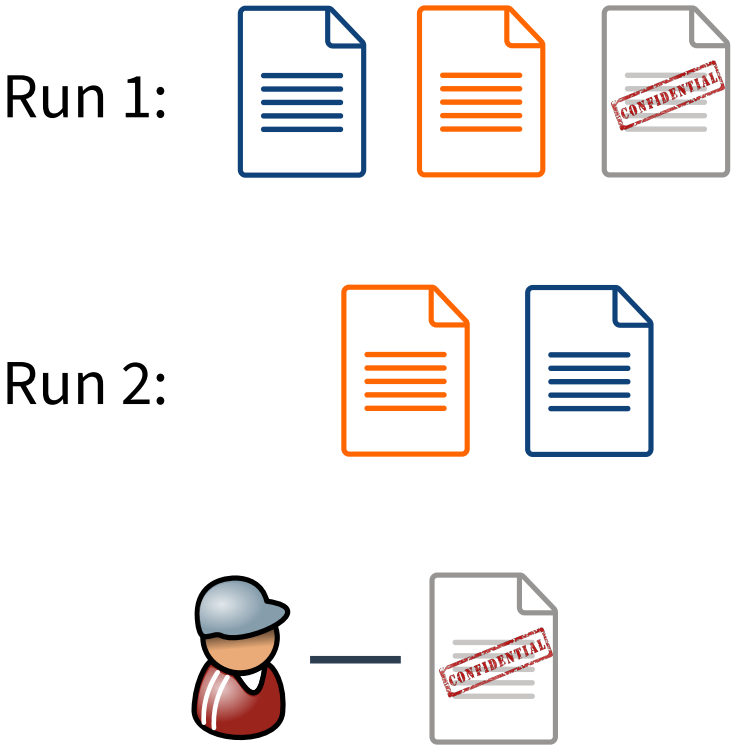
Attack on “Dissent” Protocol

- Dissent proceeds in broadcast rounds
- Outcome of the protocol is revealed to users in last broadcast
- Also network attacker sees the outcome in the last broadcast



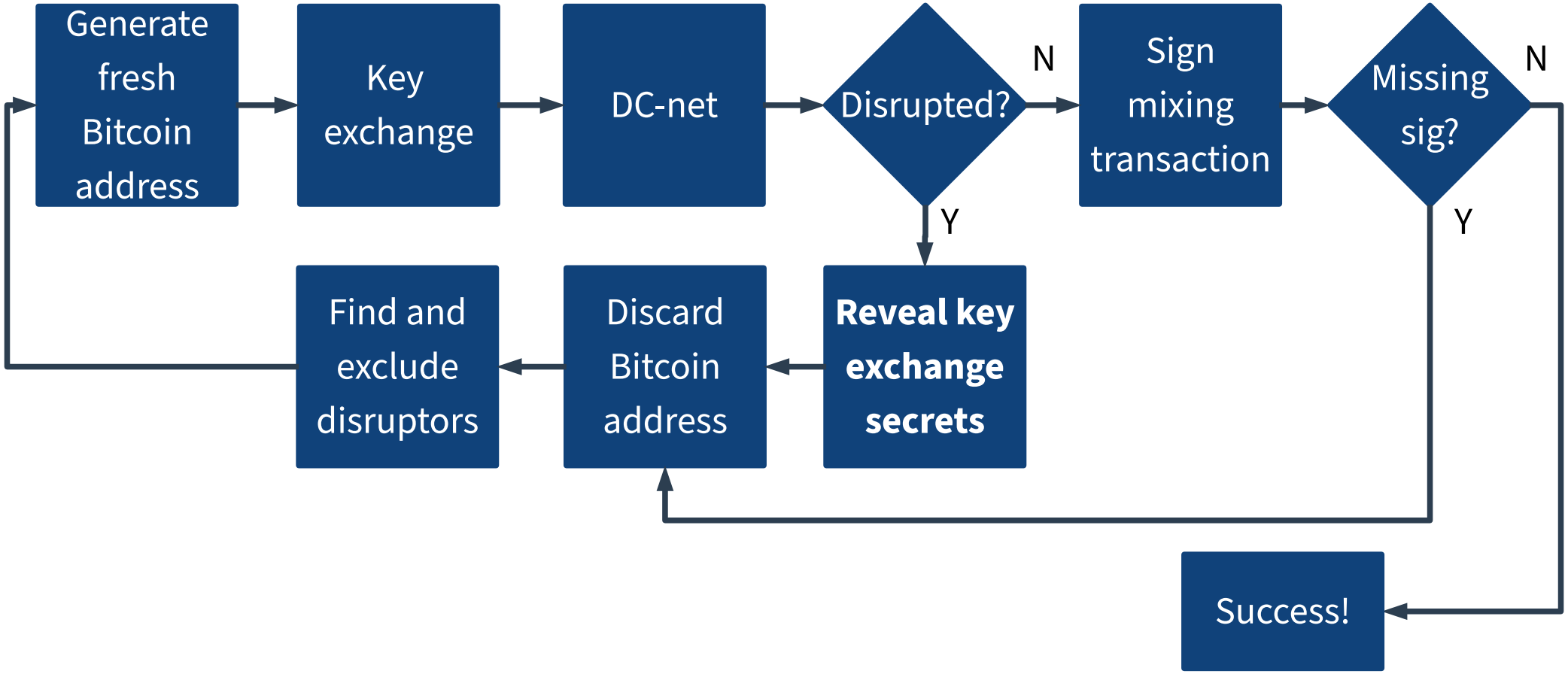
Attack on “Dissent” Protocol

- Dissent proceeds in broadcast rounds
- Outcome of the protocol is revealed to users in last broadcast
- Also network attacker sees the outcome in the last broadcast

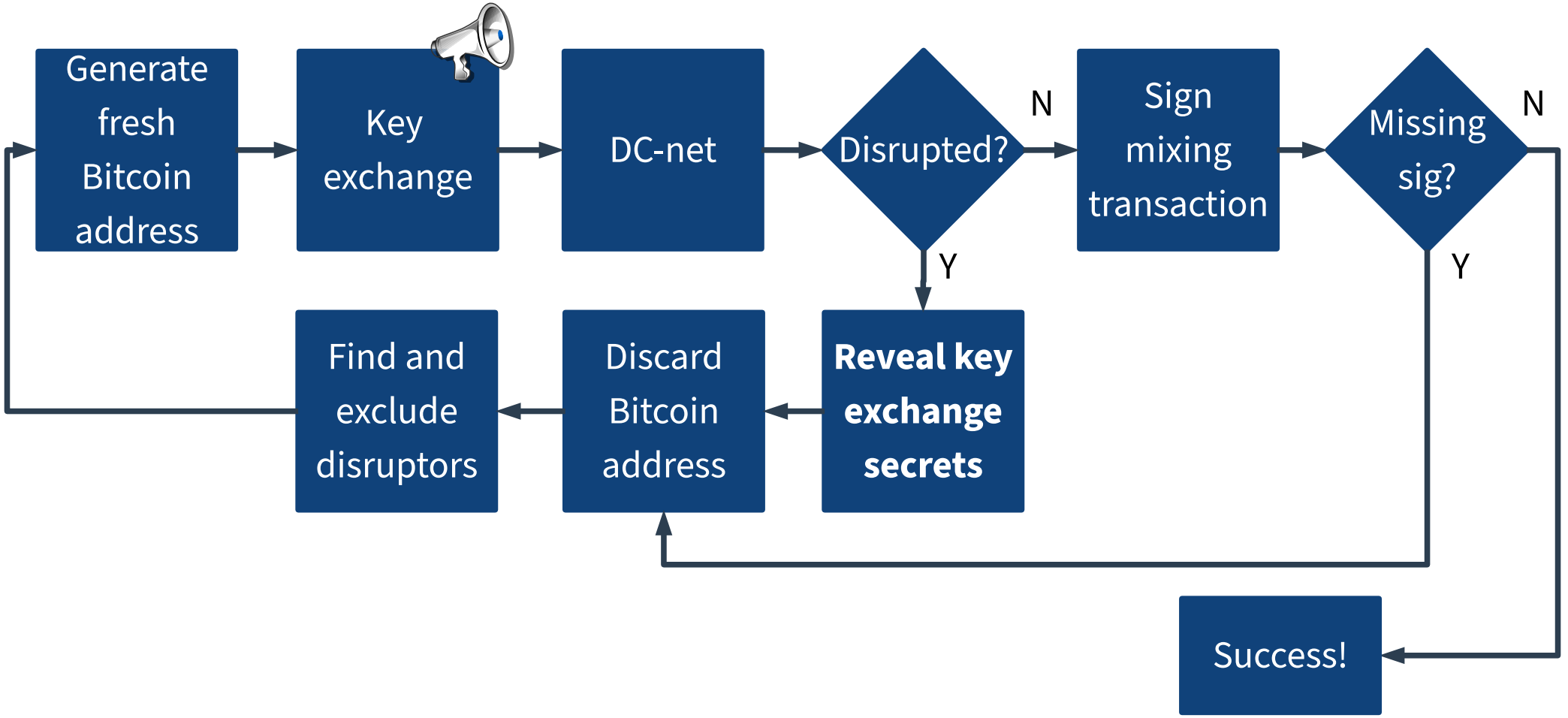


Efficiency of CoinShuffle++

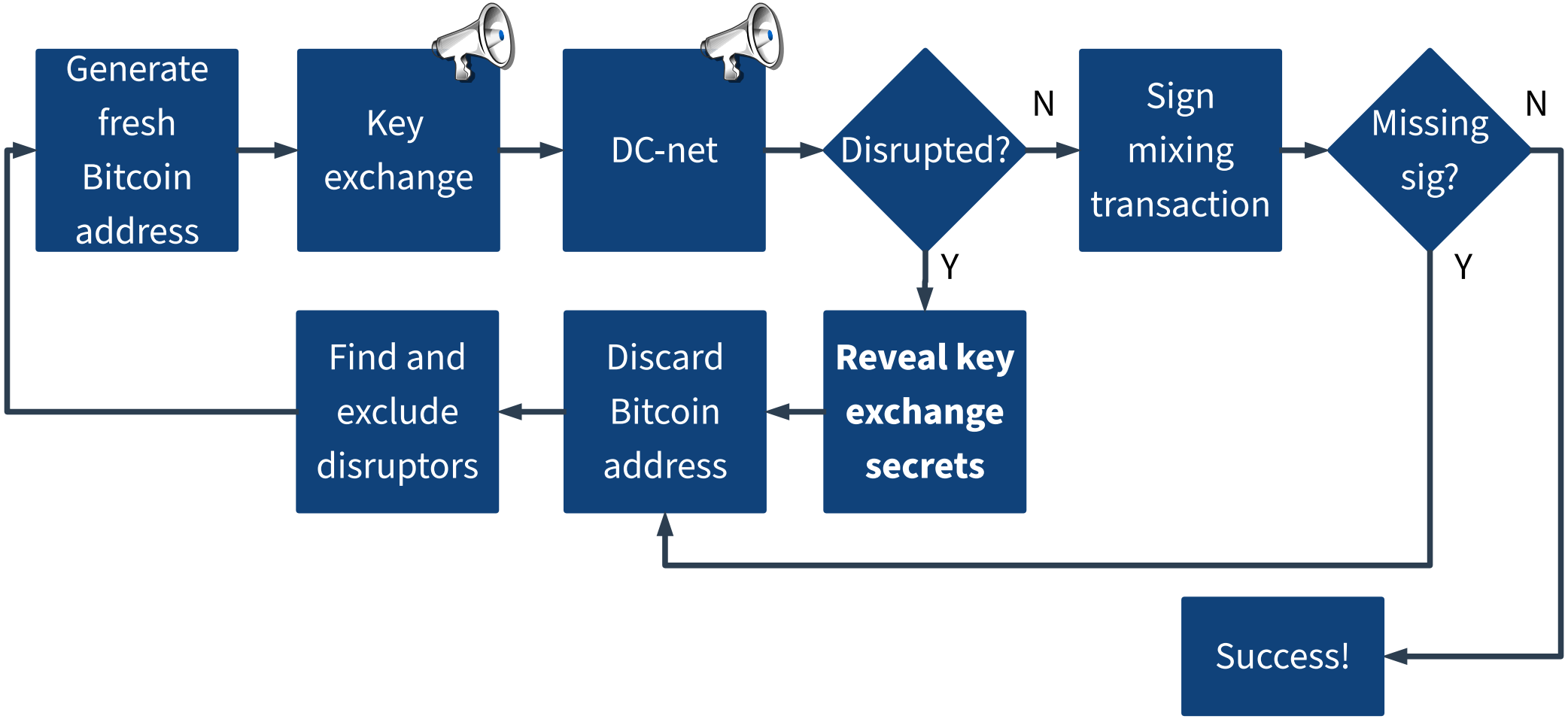
Broadcast Rounds (Naive)



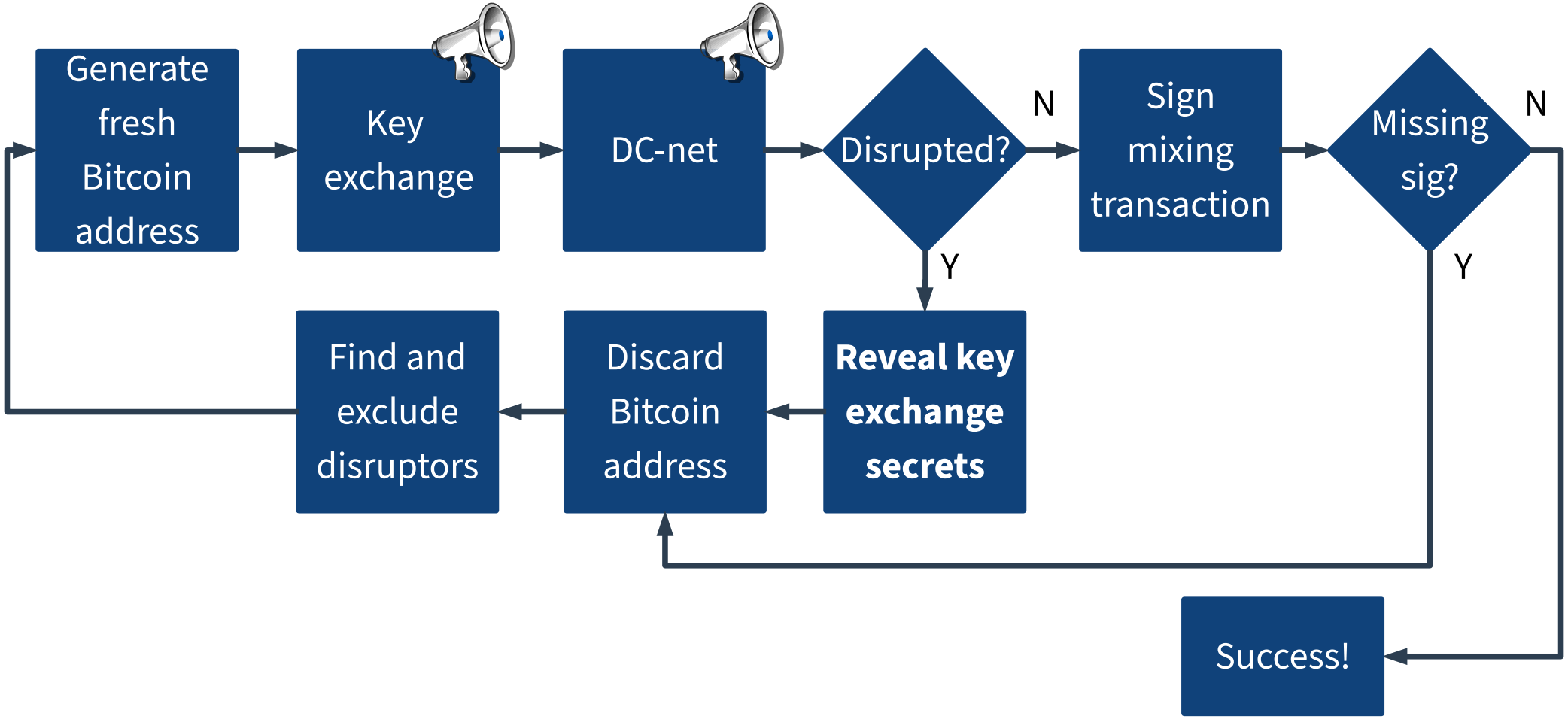
Broadcast Rounds (Naive)



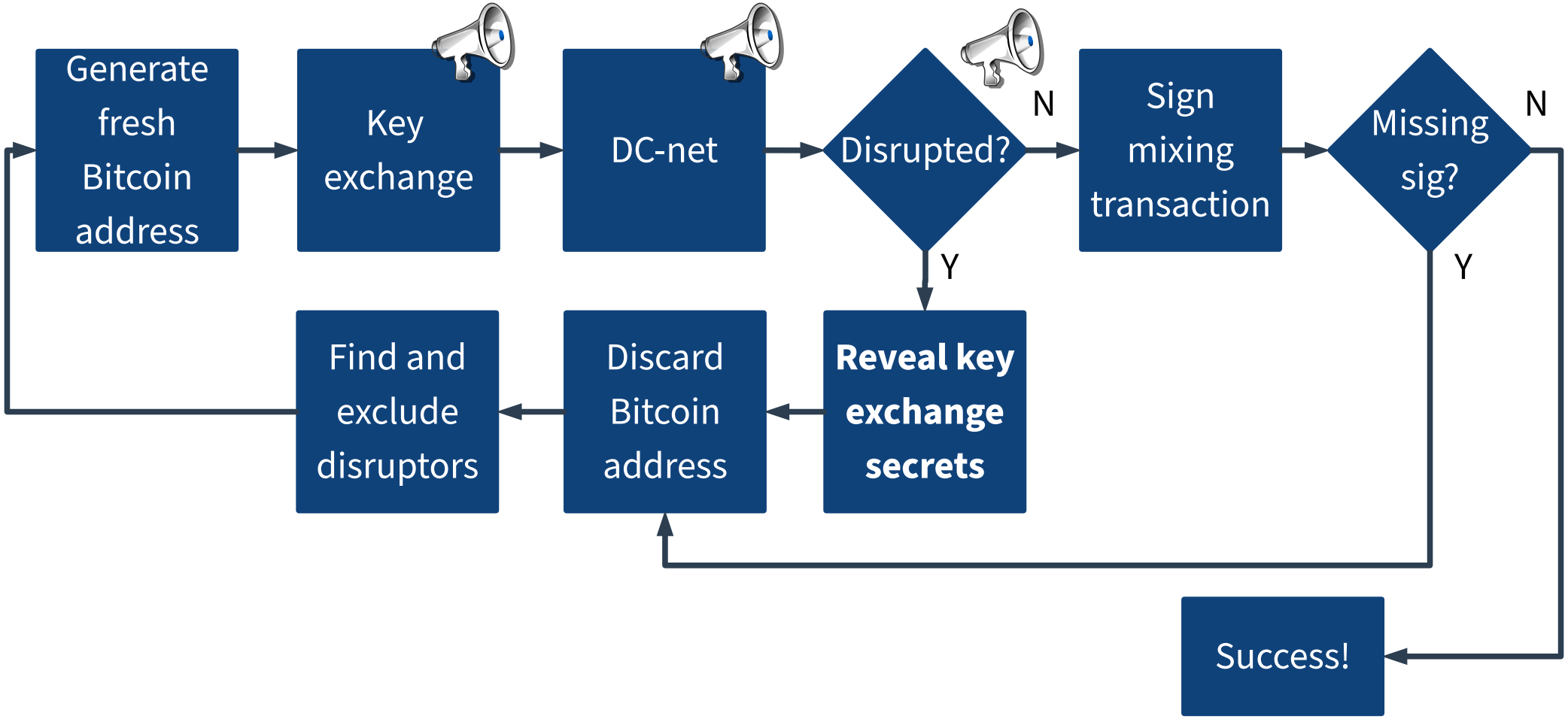
Broadcast Rounds (Naive)



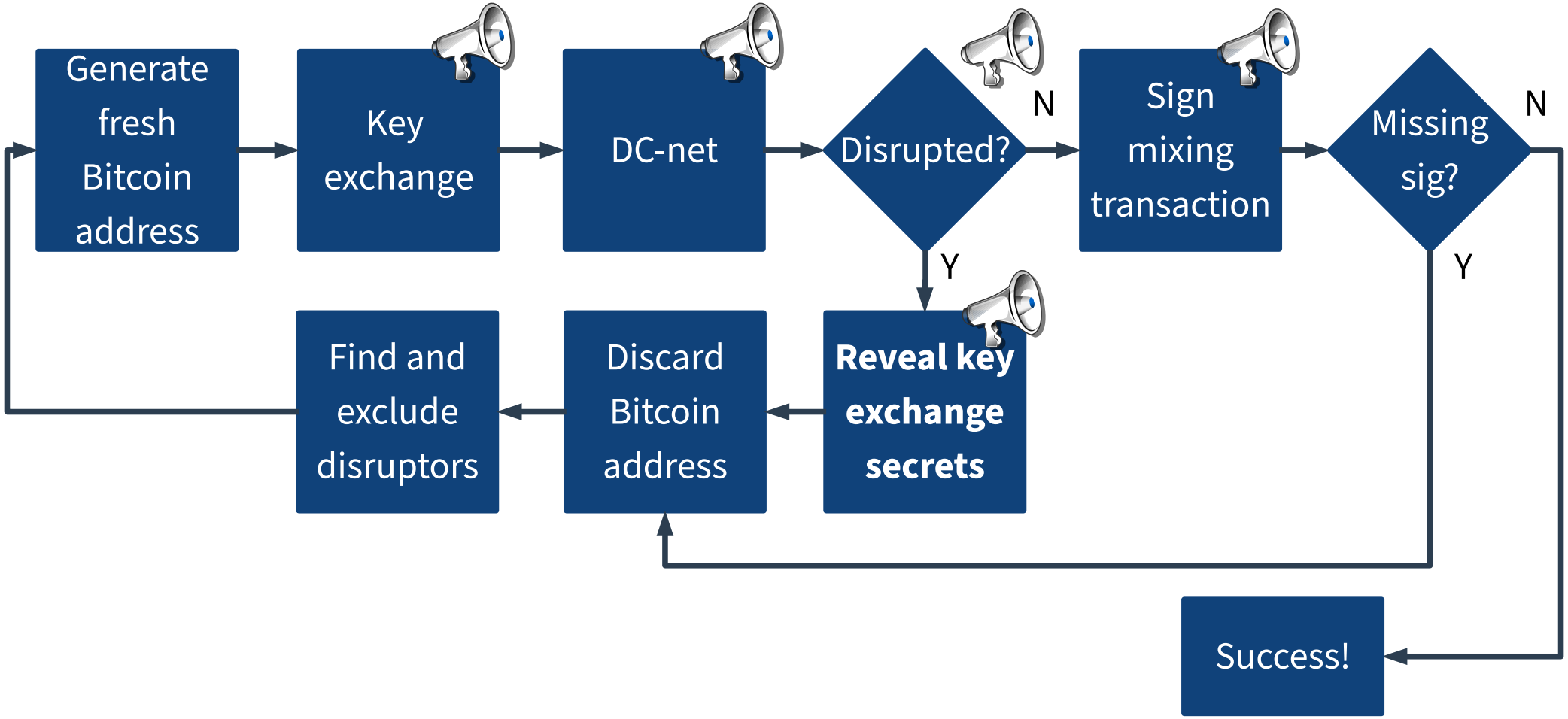
Broadcast Rounds (Naive)



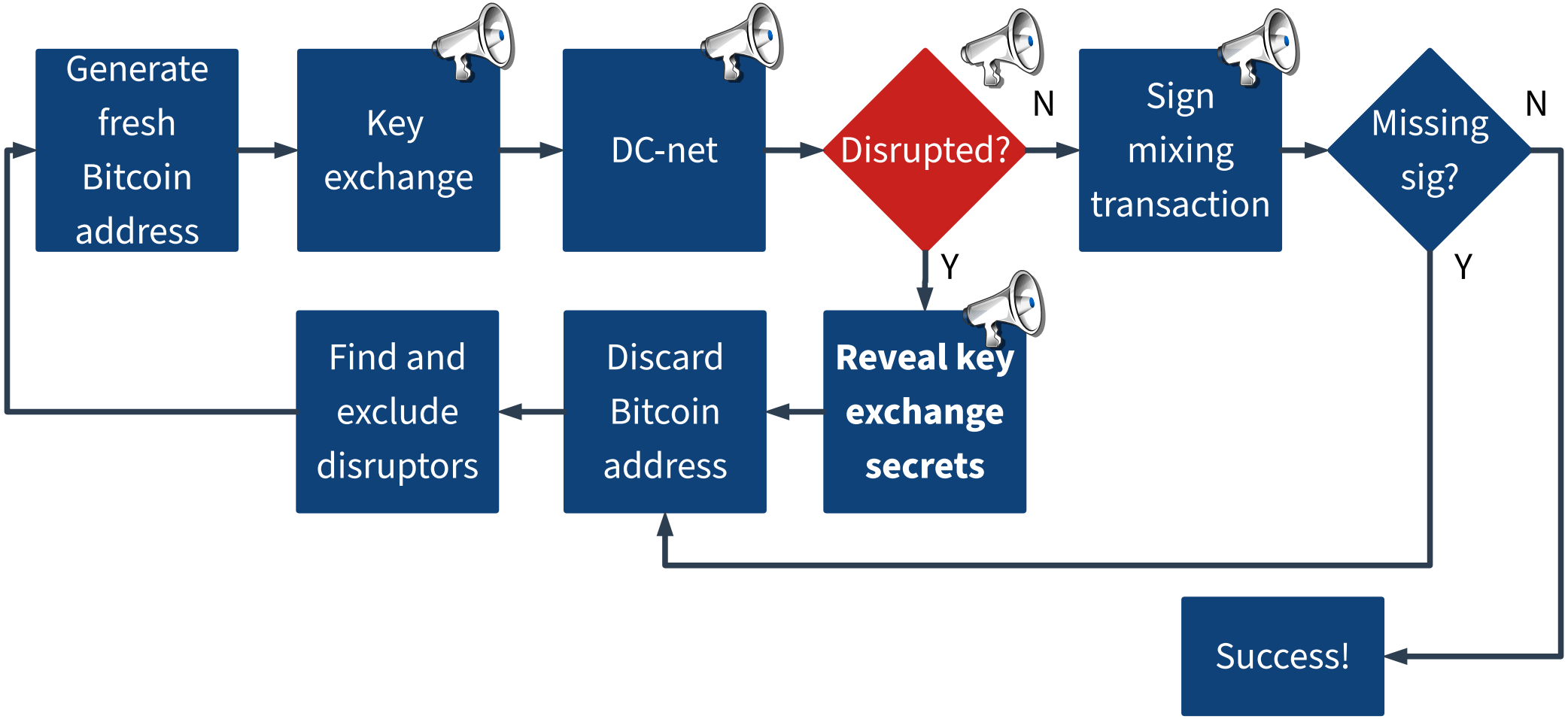
Broadcast Rounds (Naive)



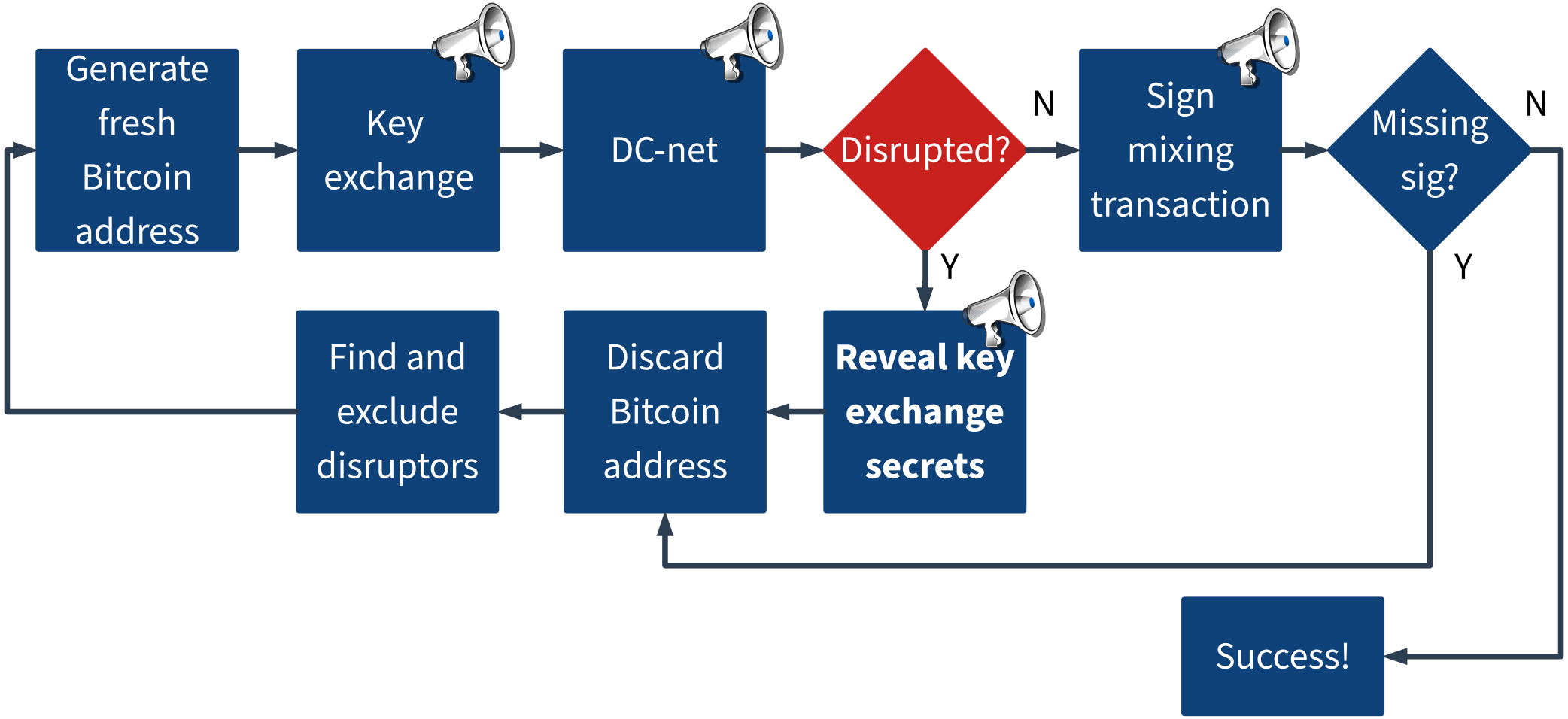
Broadcast Rounds (Naive)



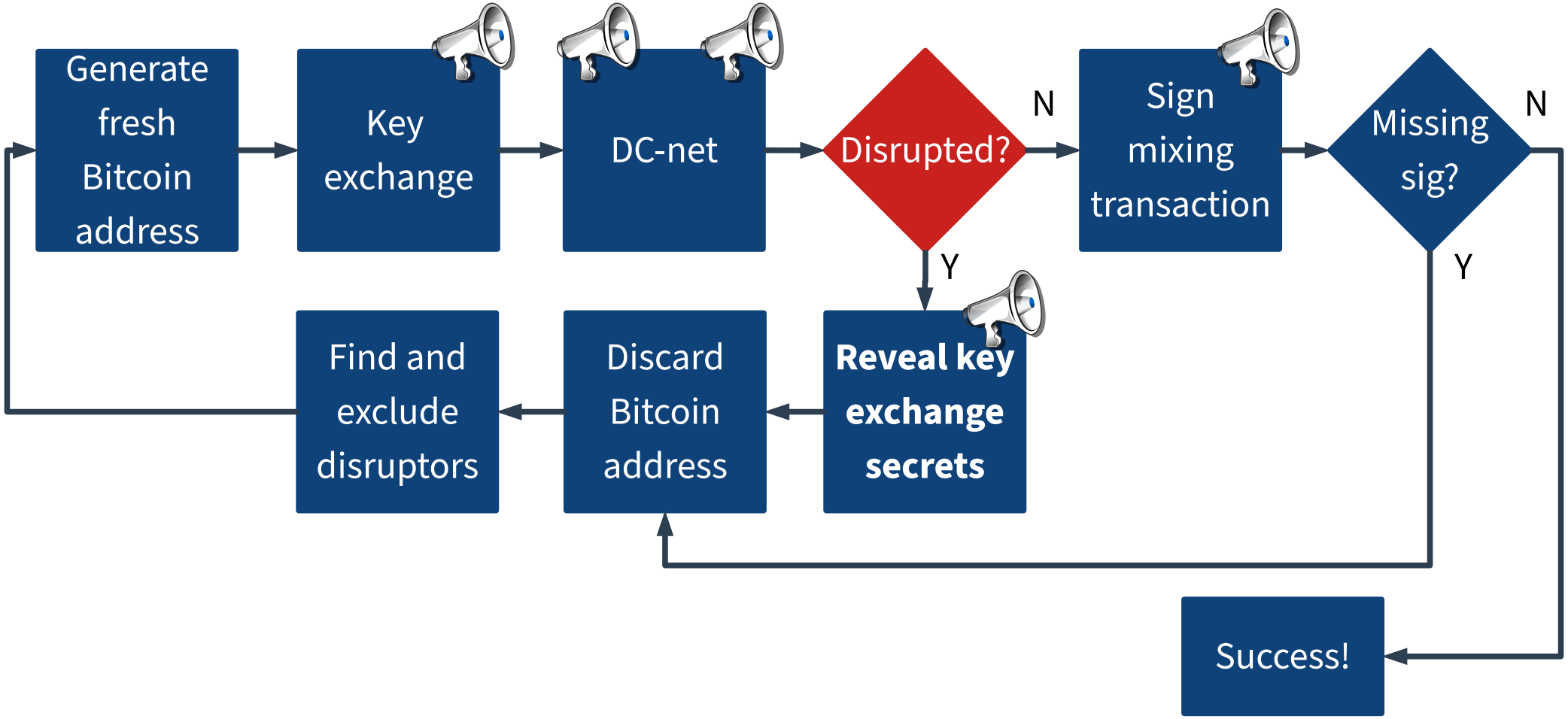
Broadcast Rounds (CoinShuffle++)



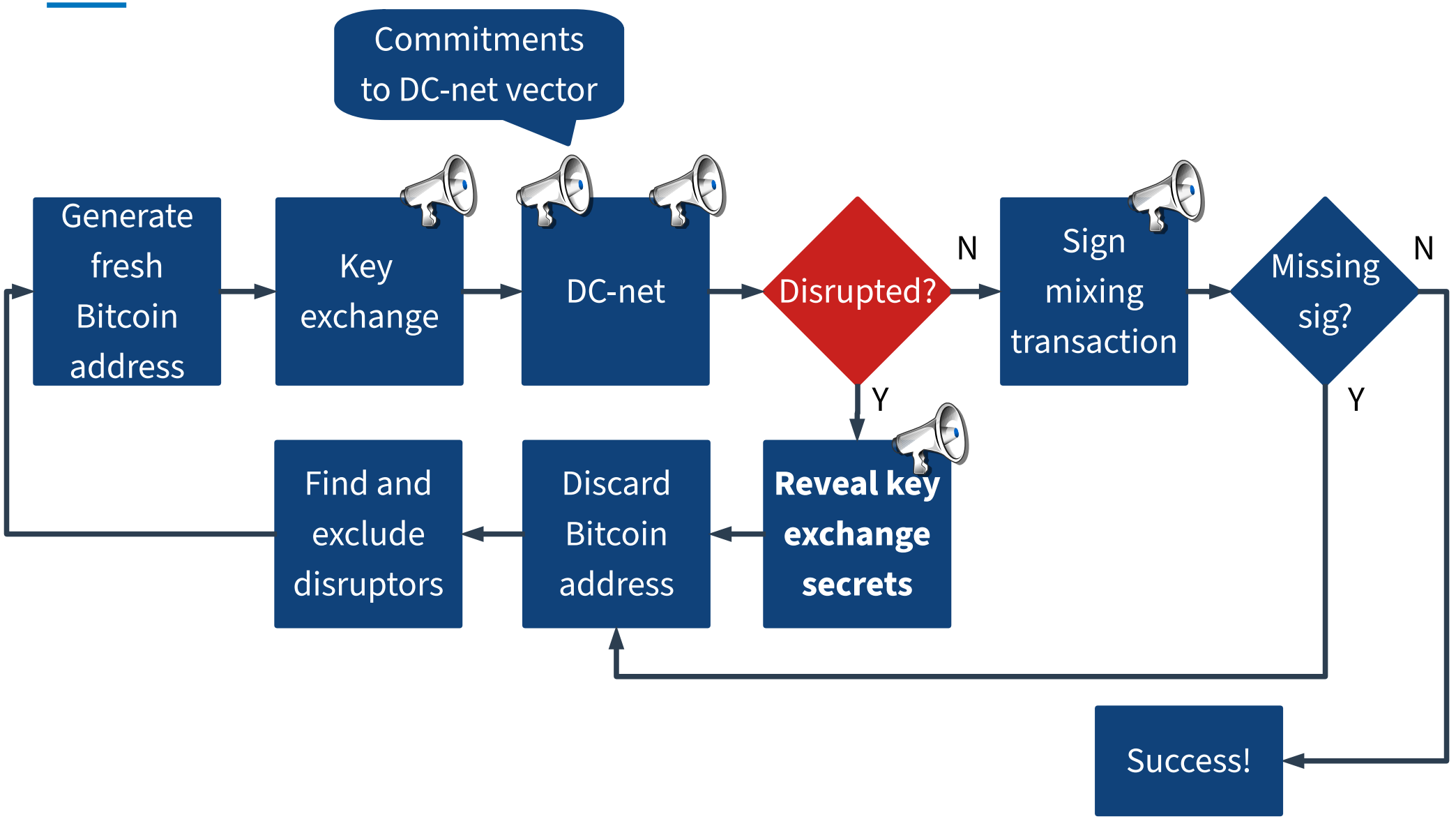
Broadcast Rounds (CoinShuffle++)



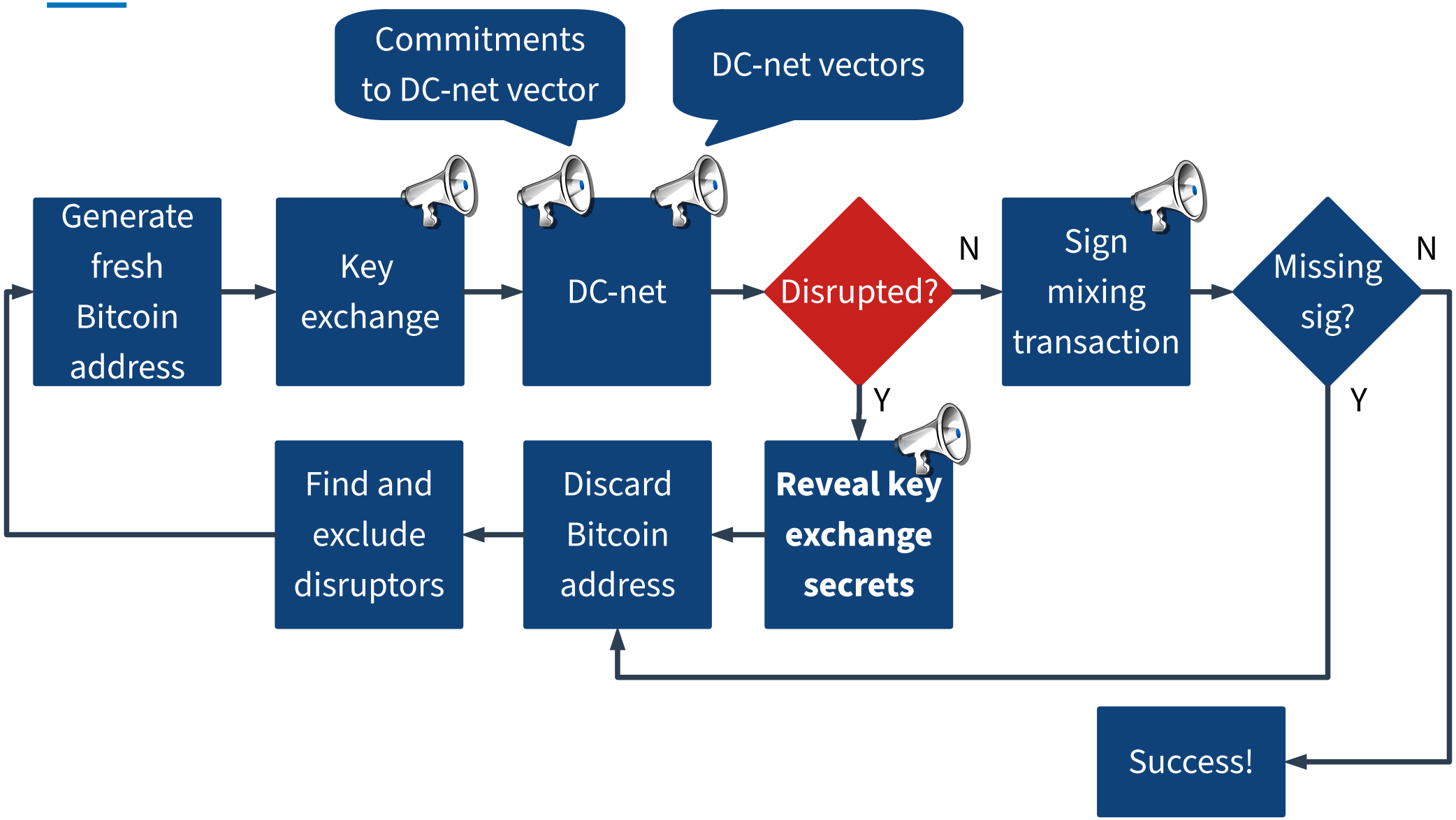
Broadcast Rounds (CoinShuffle++)



Broadcast Rounds (CoinShuffle++)



Broadcast Rounds (CoinShuffle++)



Justification (Informal)

There is a honest user whose message is disrupted.

$\Longleftrightarrow$

All honest users' messages are disrupted.

Justification (Informal)

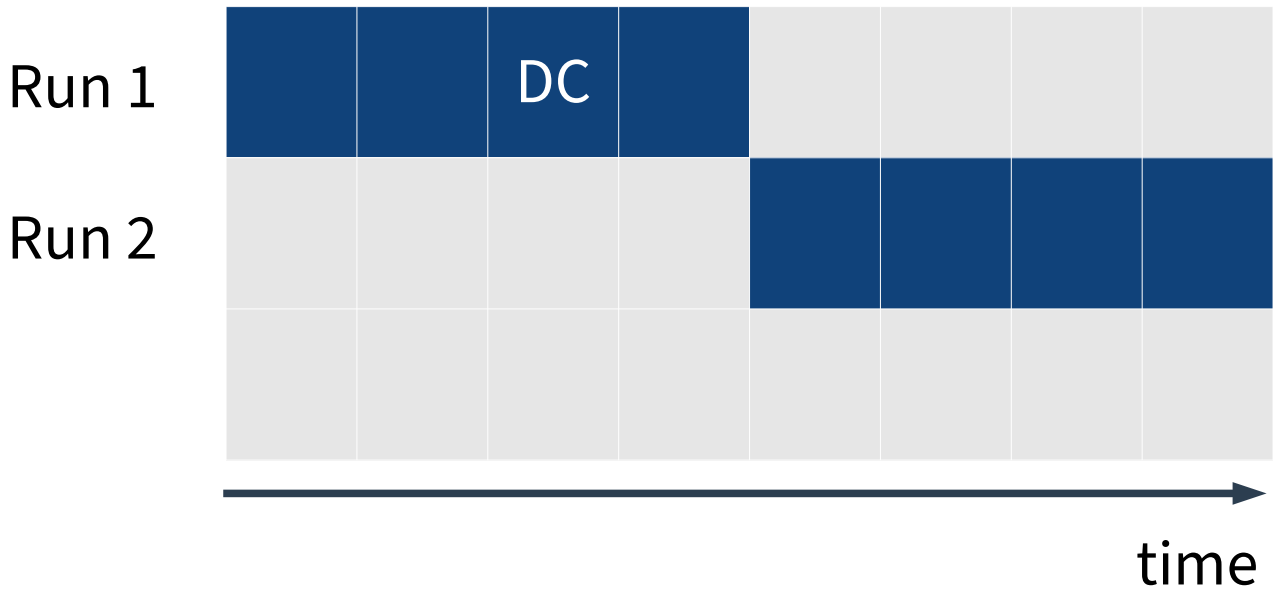
Using hash-based commitments,
in the random oracle model without further computational assumptions
against a ppt attacker, with overwhelming probability:

There is a honest user whose message is disrupted.

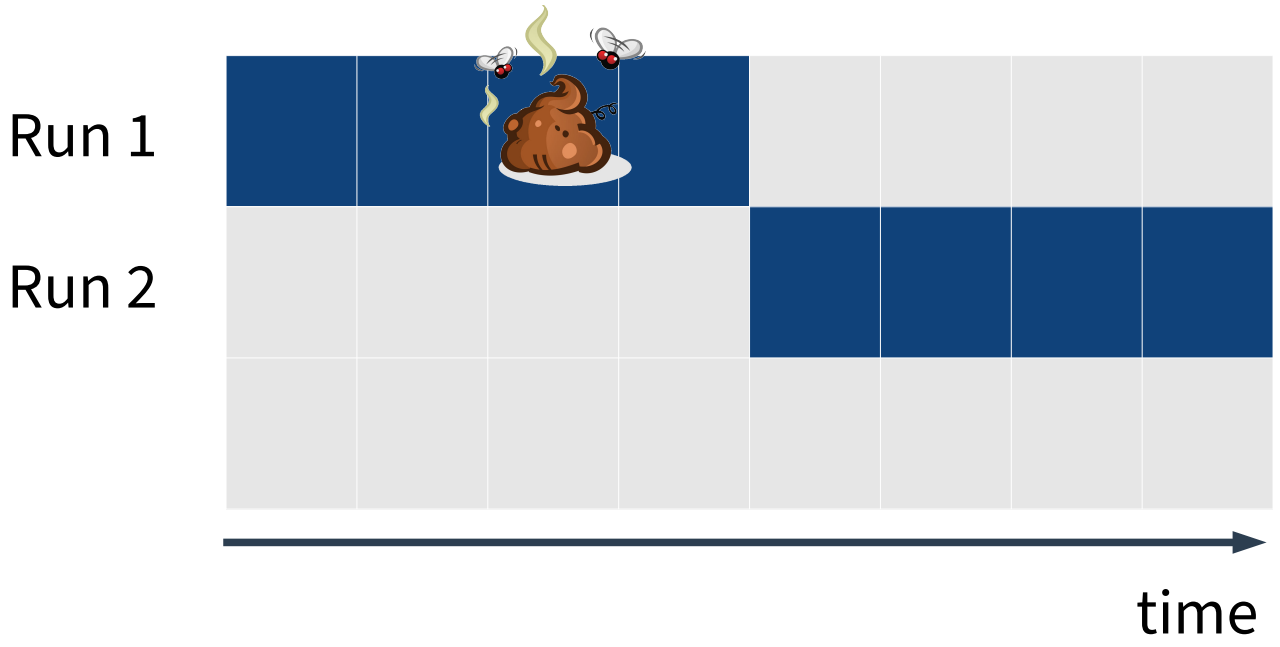
$\Longleftrightarrow$

All honest users' messages are disrupted.

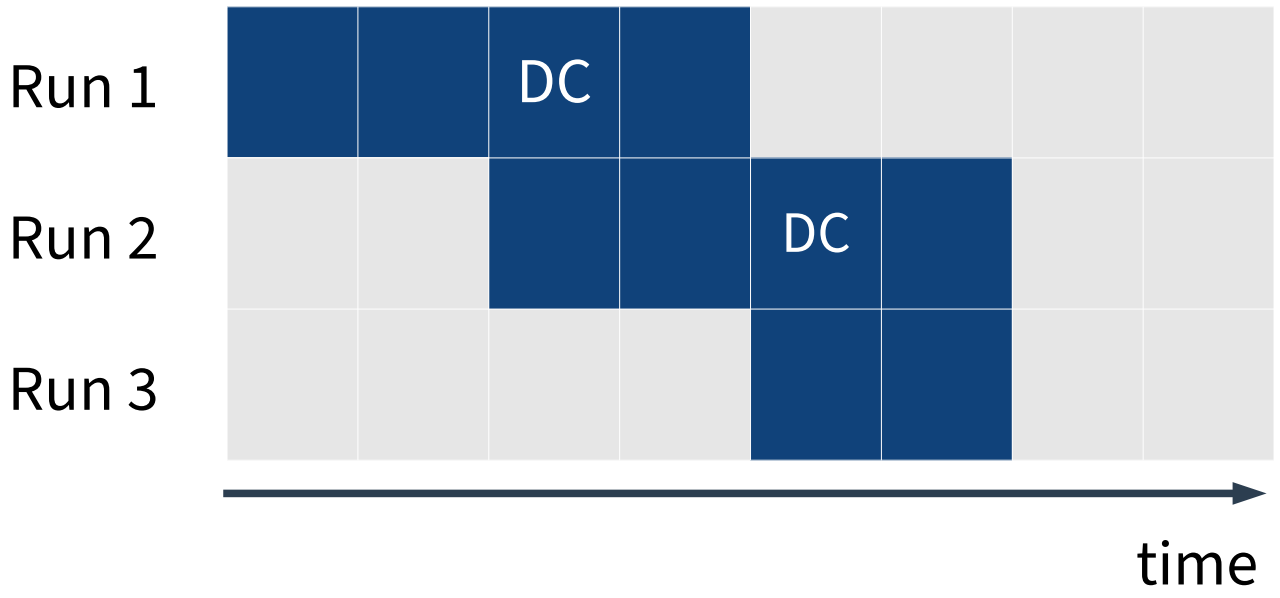
Example Execution (simple)



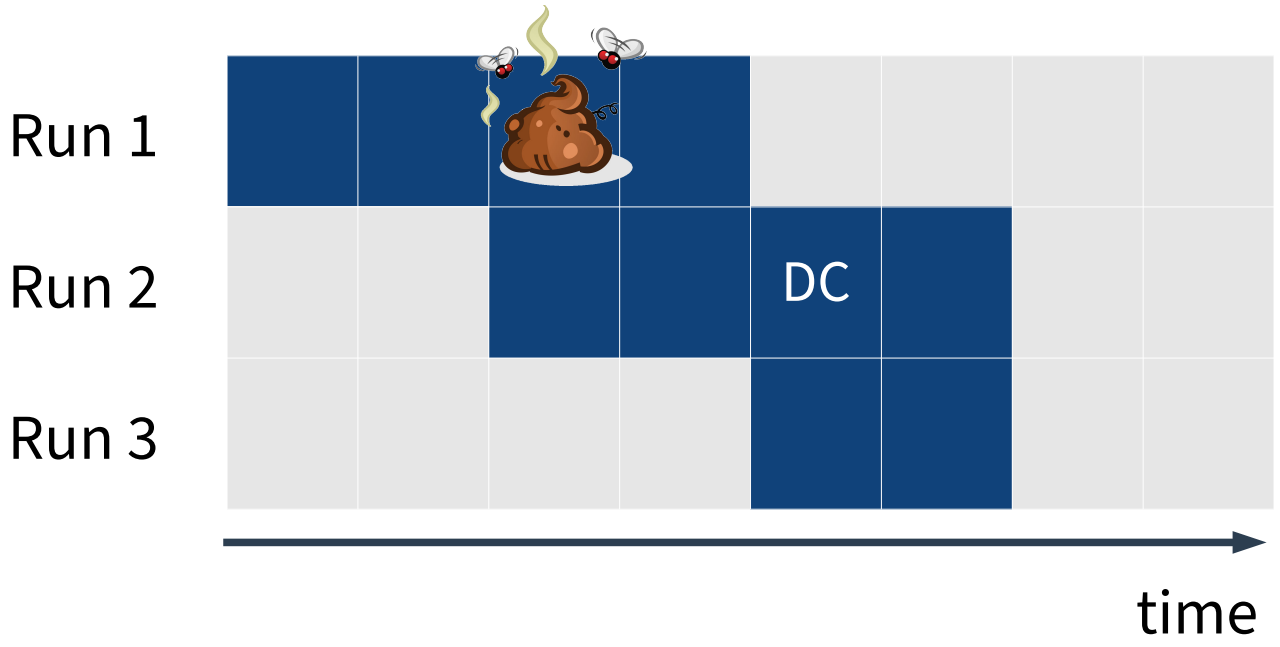
Example Execution (simple)



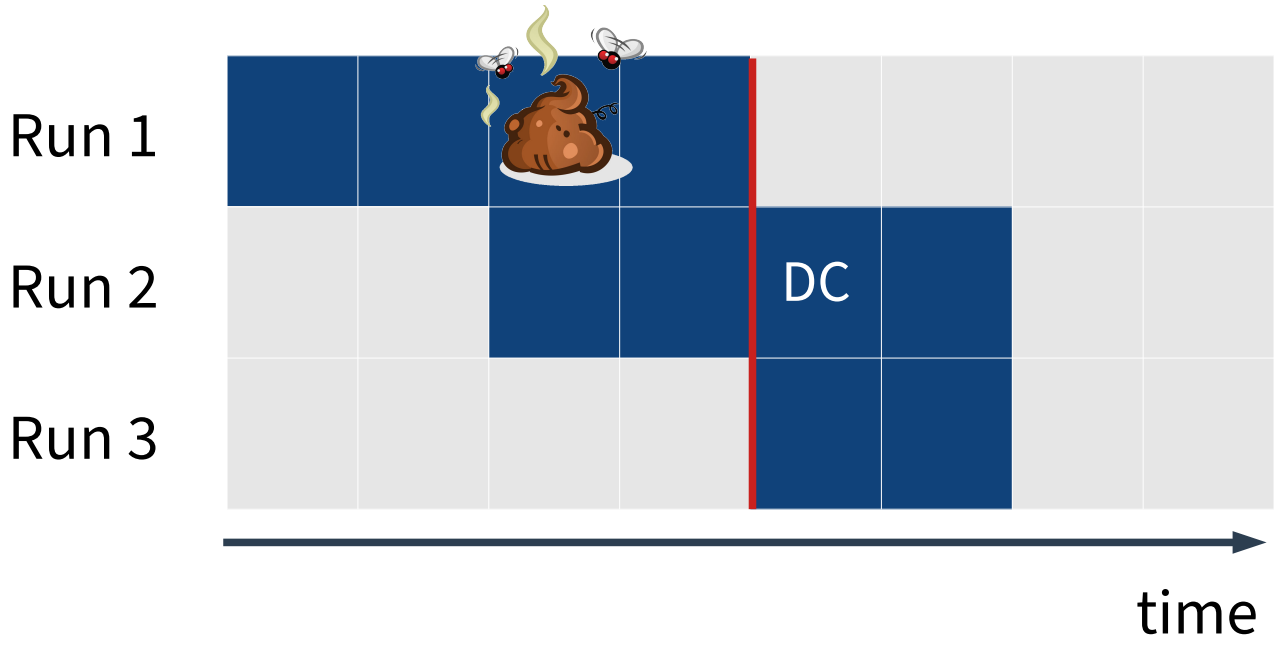
Example Execution (CoinShuffle++)



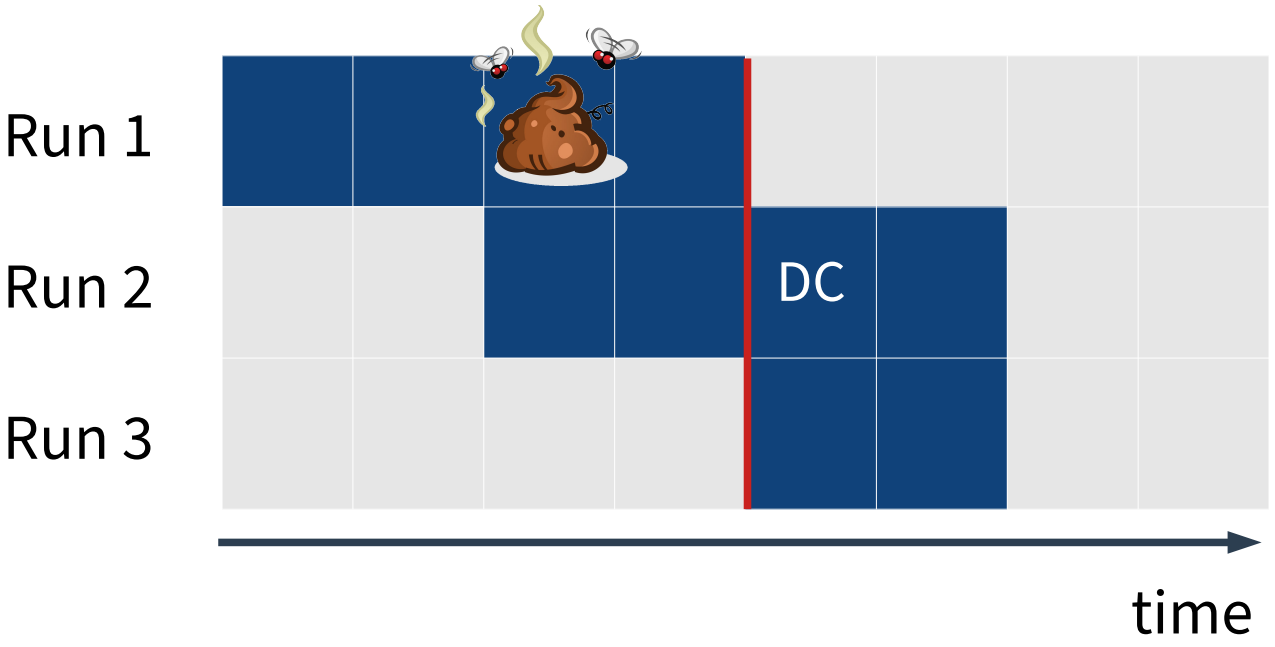
Example Execution (CoinShuffle++)



Example Execution (CoinShuffle++)

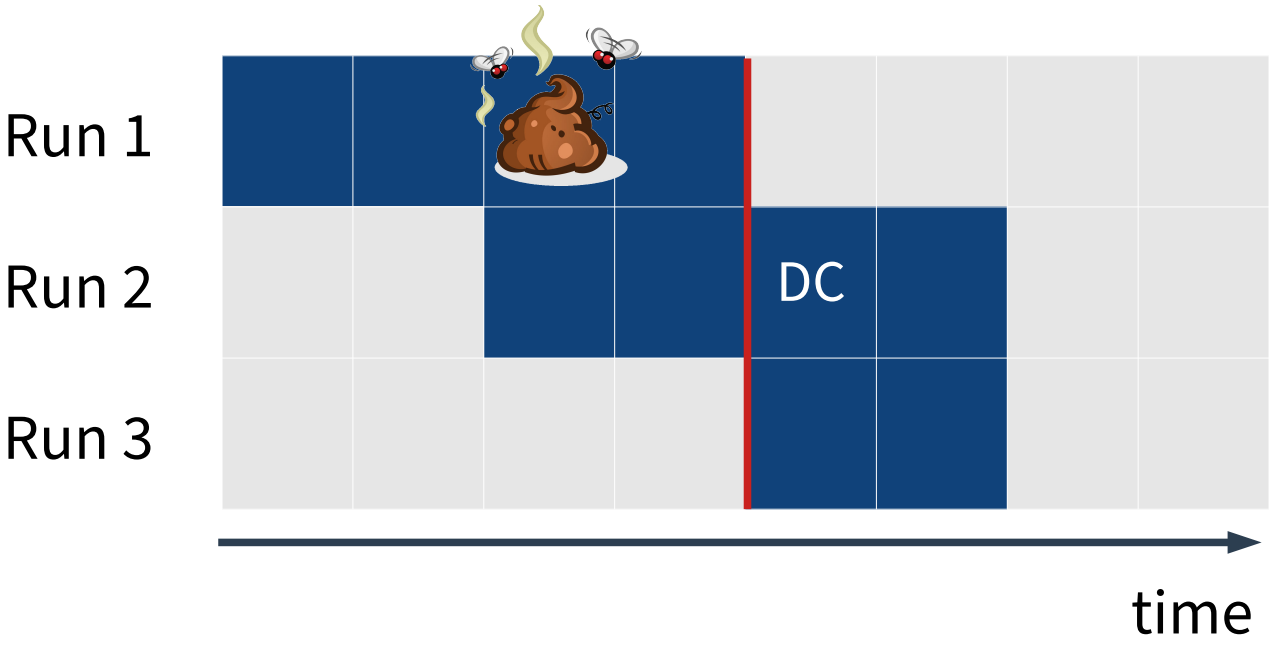


Example Execution (CoinShuffle++)



Let *f* be the number of malicious users.

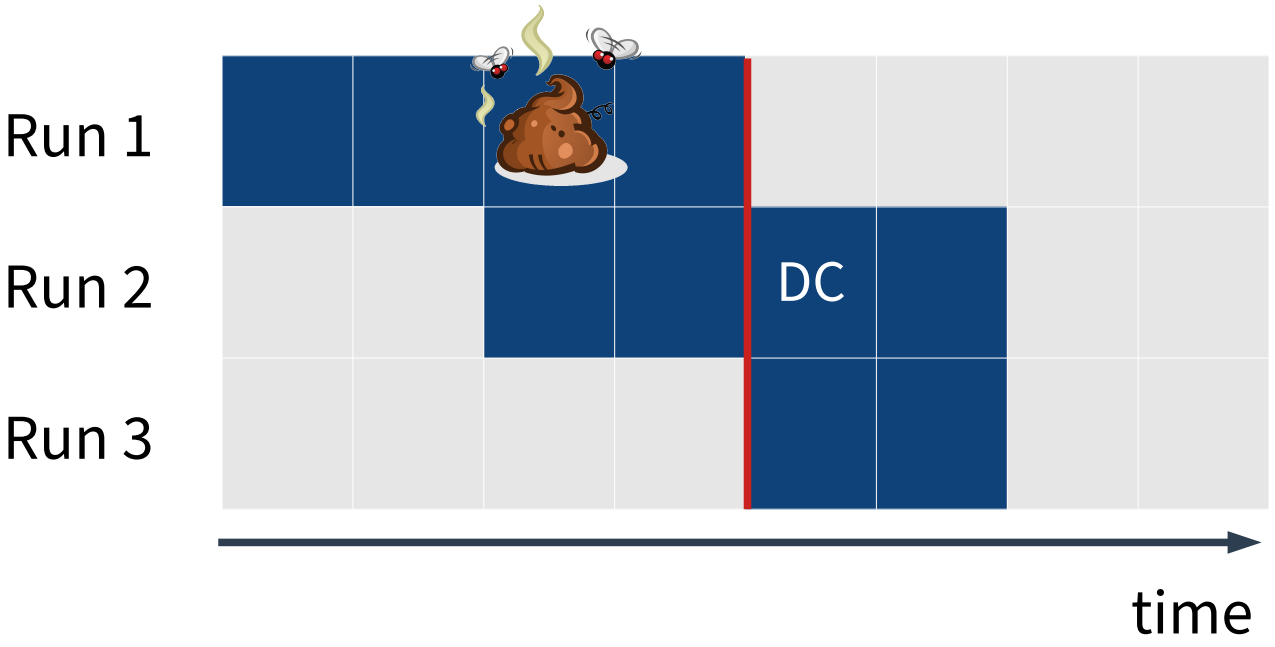
Example Execution (CoinShuffle++)



Let f be the number of malicious users.

CoinShuffle++ needs $4 + 2f$ broadcast rounds.

Example Execution (CoinShuffle++)



Let f be the number of malicious users.

CoinShuffle++ needs $4 + 2f$ broadcast rounds.

Previous work: $O(nf)$

Practical Evaluation

Practical Evaluation

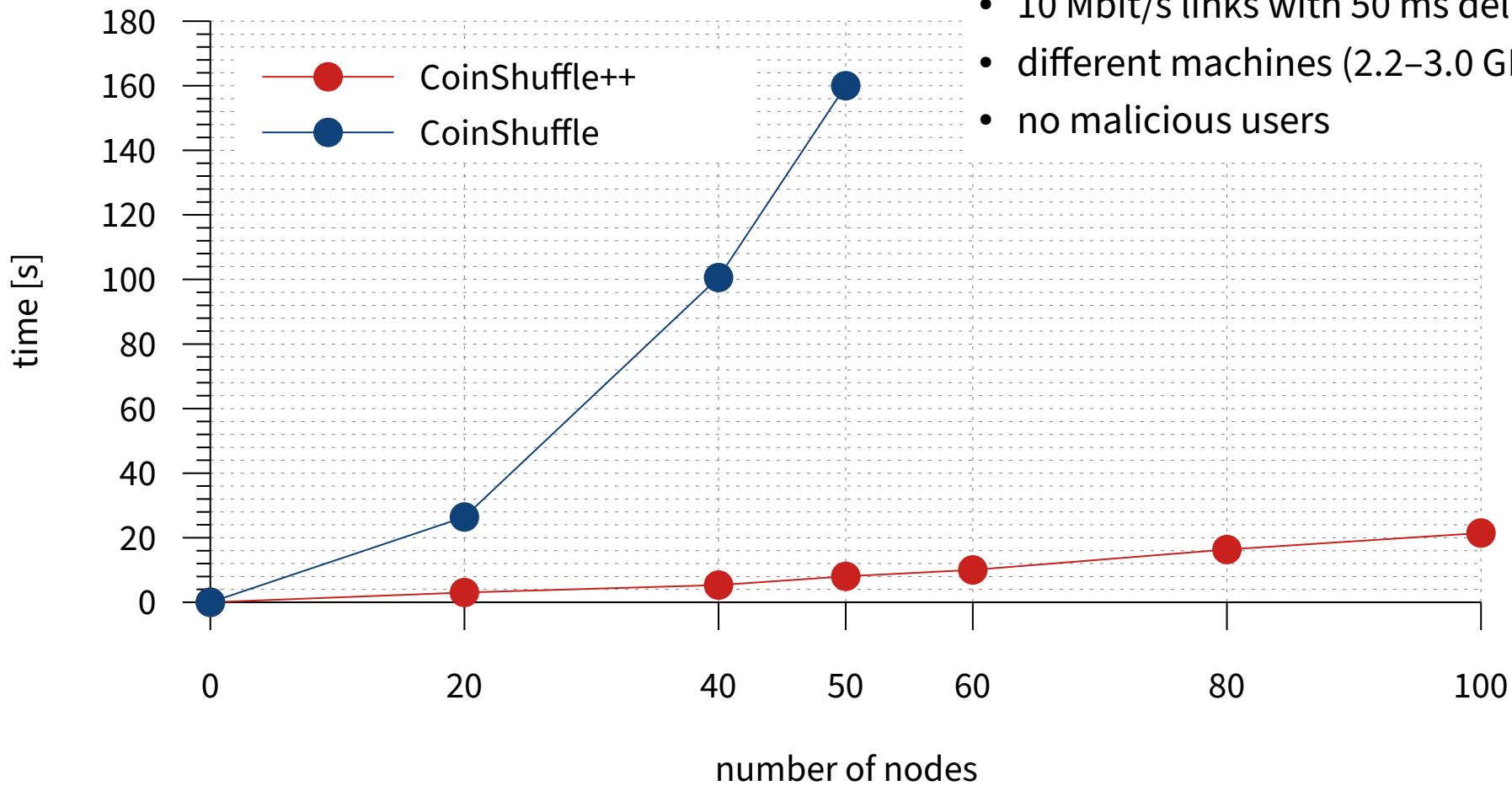
Network Setup

- 10 Mbit/s links with 50 ms delay
- different machines (2.2–3.0 GHz)
- no malicious users

Practical Evaluation

Network Setup

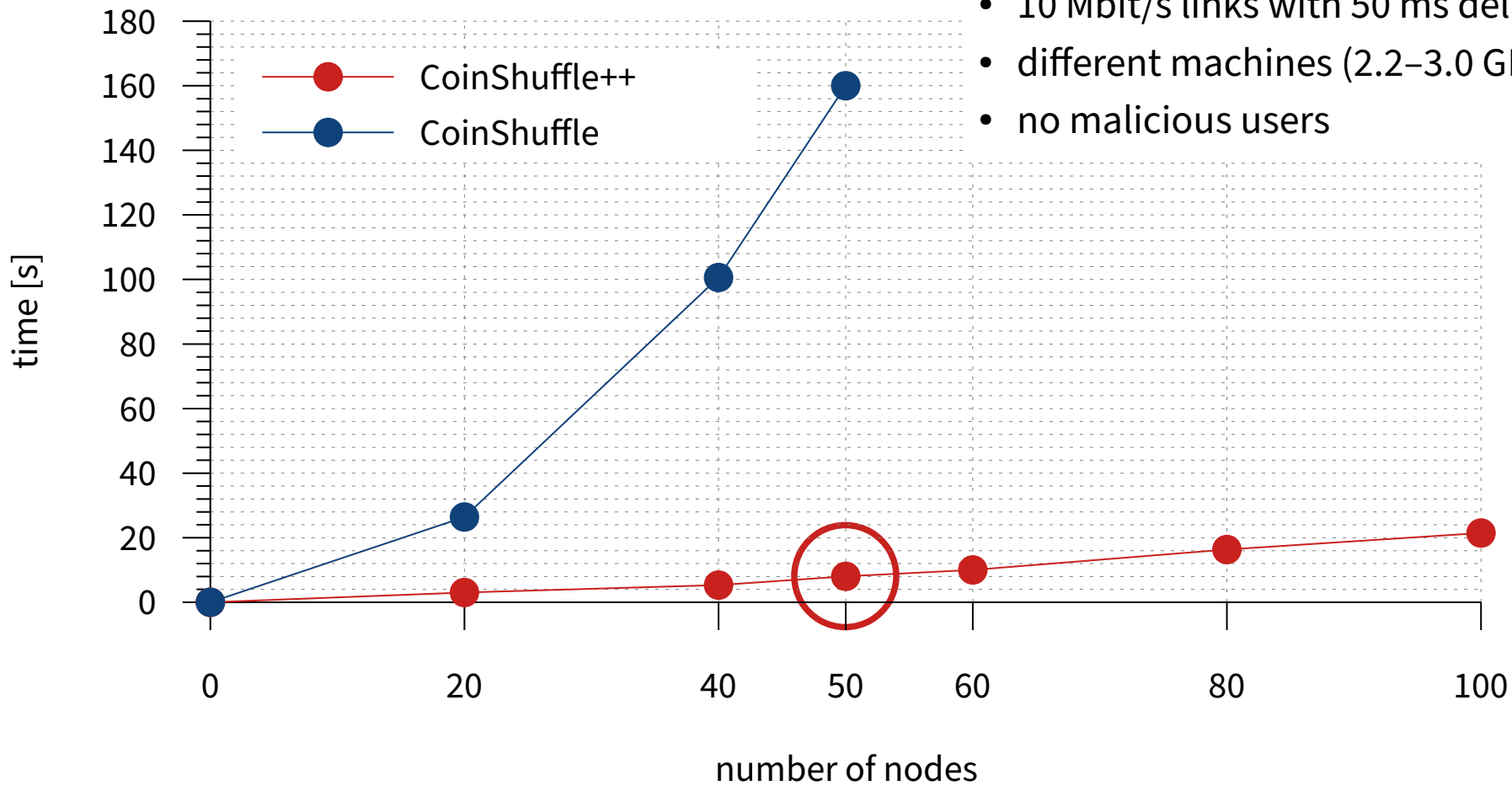
- 10 Mbit/s links with 50 ms delay
- different machines (2.2–3.0 GHz)
- no malicious users



Practical Evaluation

Network Setup

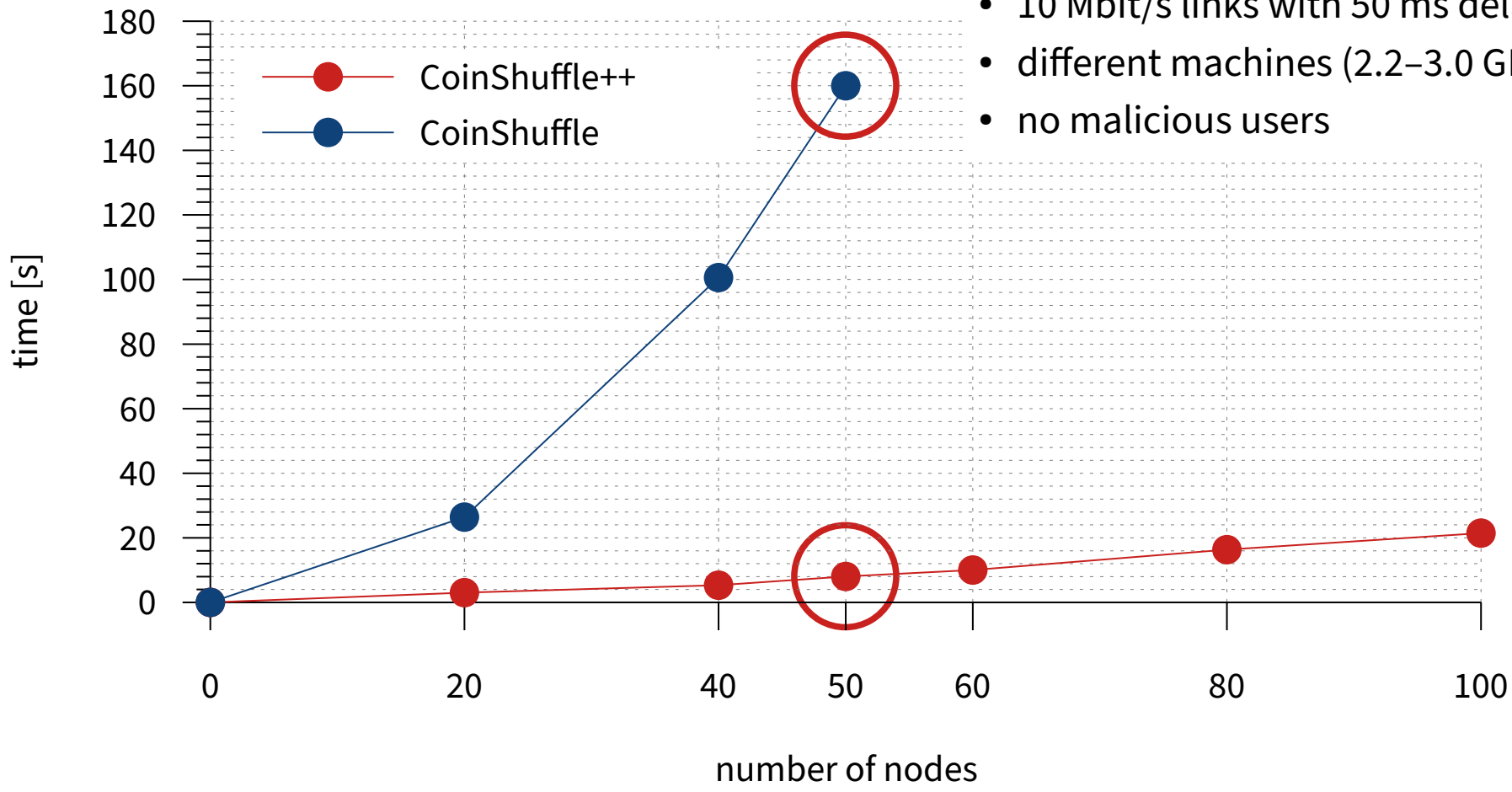
- 10 Mbit/s links with 50 ms delay
- different machines (2.2–3.0 GHz)
- no malicious users



Practical Evaluation

Network Setup

- 10 Mbit/s links with 50 ms delay
- different machines (2.2–3.0 GHz)
- no malicious users

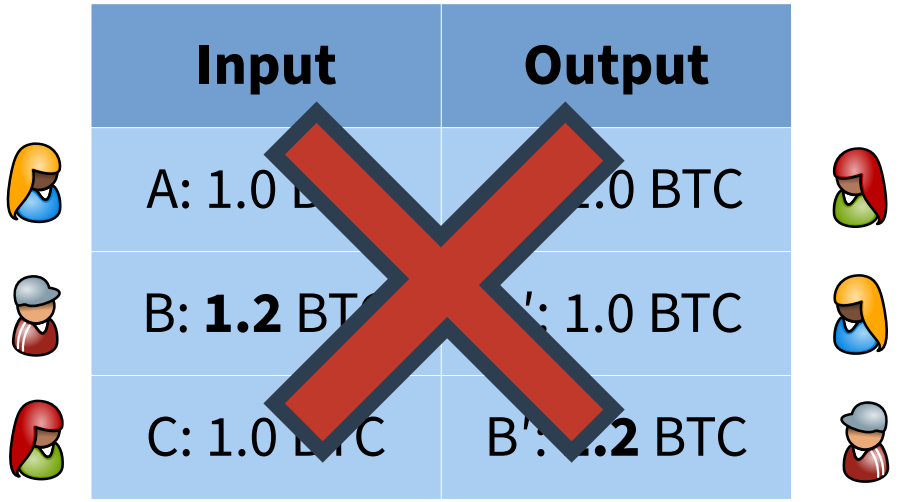


Limitations of CoinShuffle++

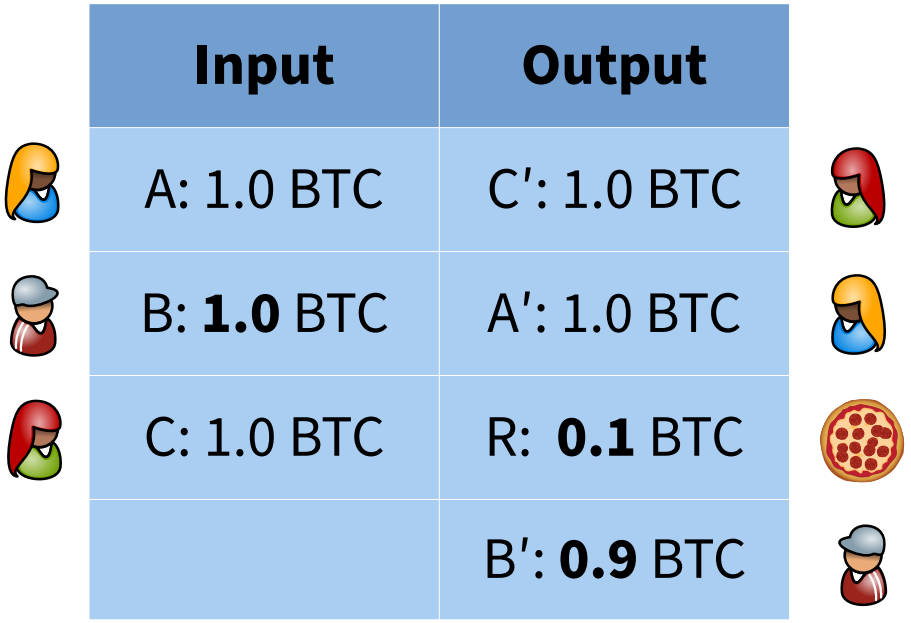
1. Handling Unequal Inputs

	Input	Output	
	A: 1.0 BTC	C′: 1.0 BTC	
	B: **1.2** BTC	A′: 1.0 BTC	
	C: 1.0 BTC	B′: **1.2** BTC	

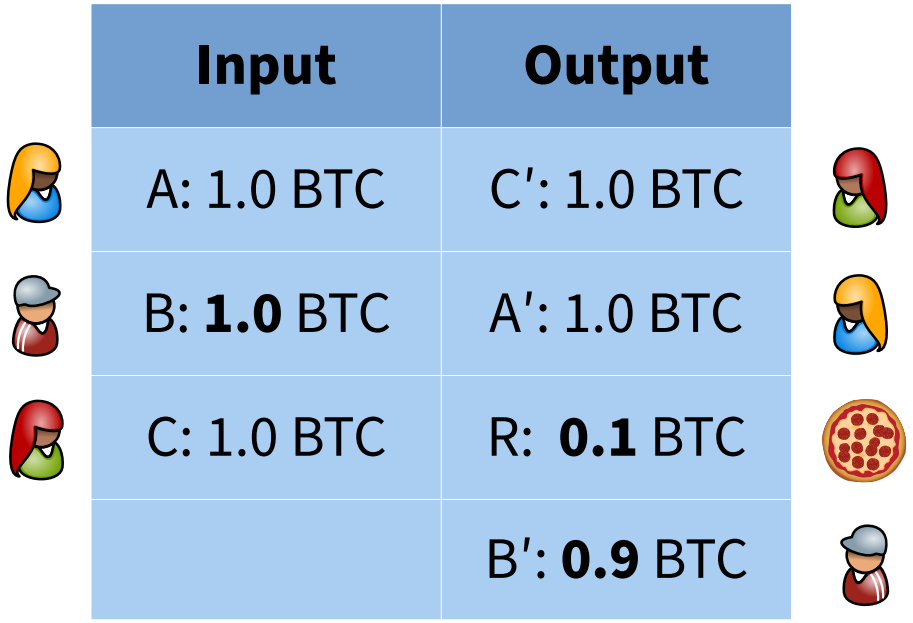
1. Handling Unequal Inputs



2. Mixing and Paying Simultaneously



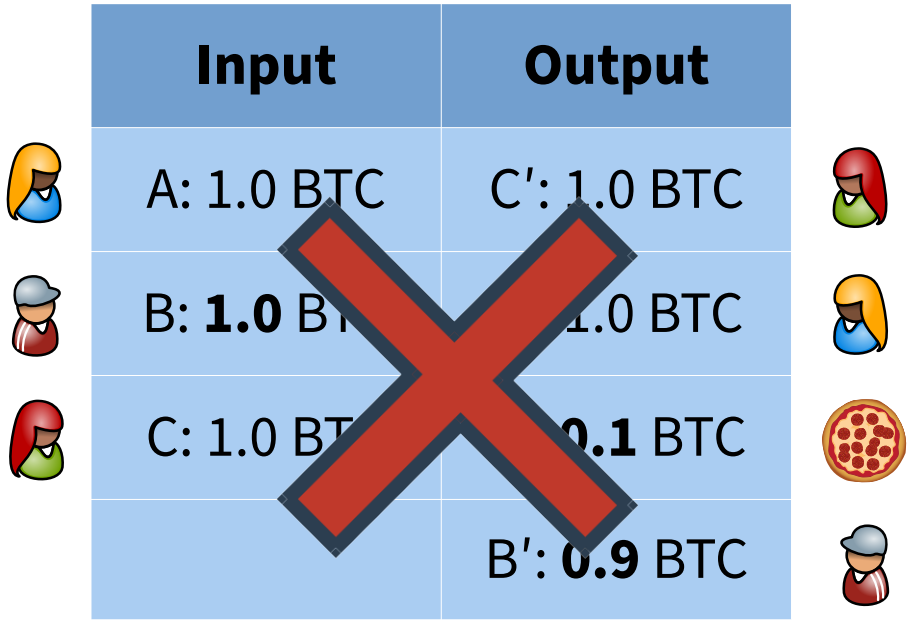
2. Mixing and Paying Simultaneously



One of Bob's input messages to P2P mixing protocol:

(R, 0.1)

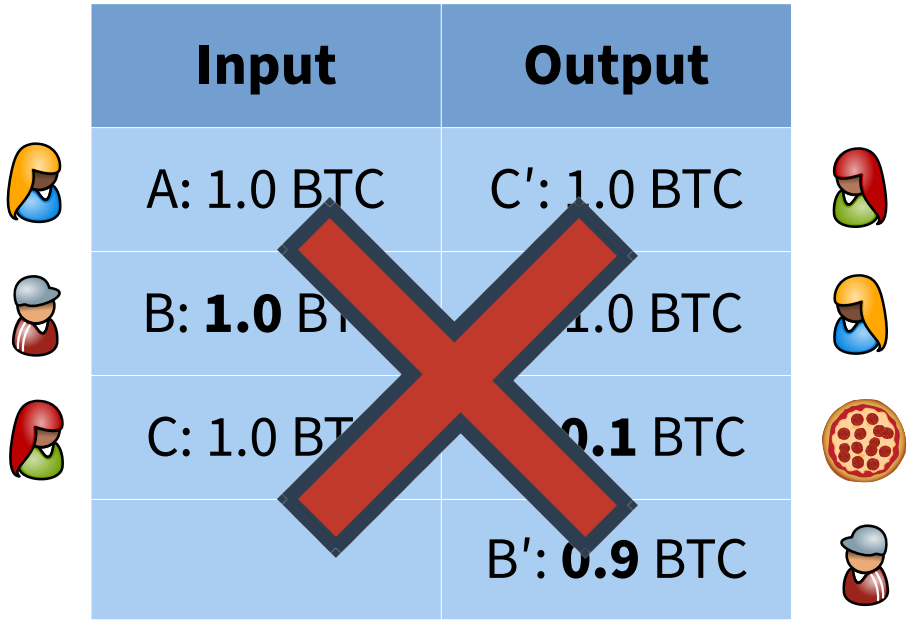
2. Mixing and Paying Simultaneously



One of Bob's input messages to P2P mixing protocol:

(R, 0.1)

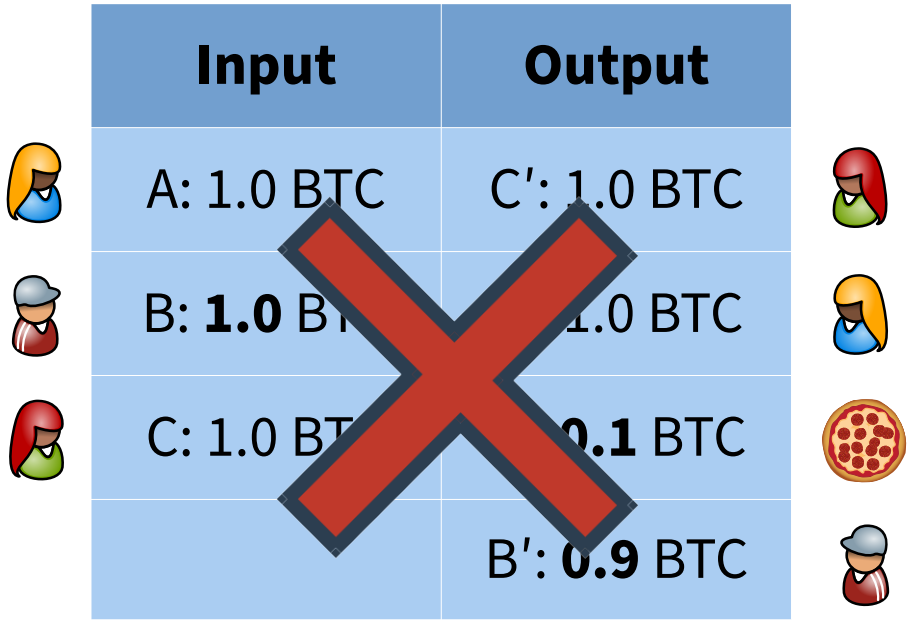
2. Mixing and Paying Simultaneously



One of Bob's input messages to P2P mixing protocol:

(R, 0.1)

2. Mixing and Paying Simultaneously

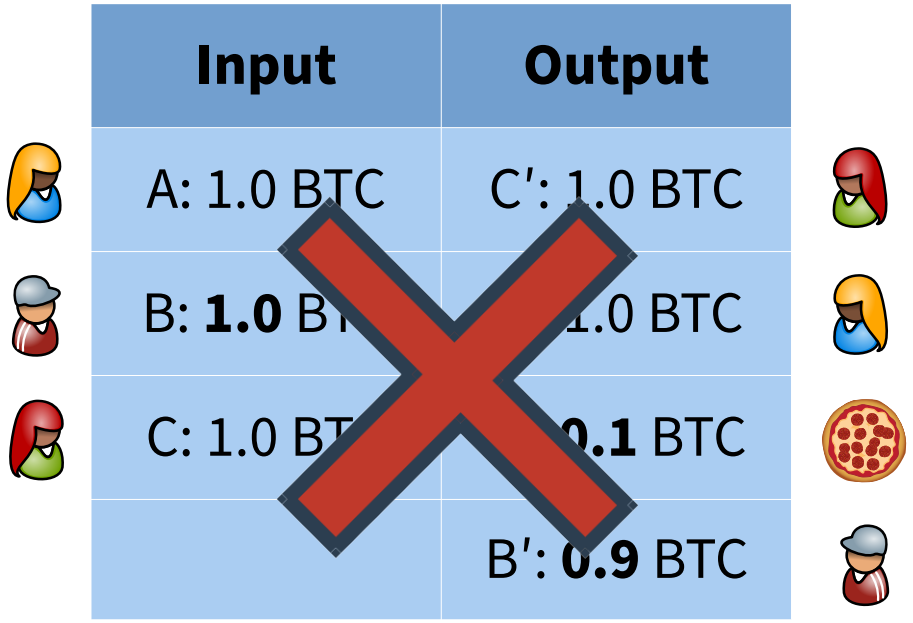


One of Bob's input messages to P2P mixing protocol:

(R, 0.1)

Fixed message!

2. Mixing and Paying Simultaneously



One of Bob's input messages to P2P mixing protocol:

(R, 0.1)

Fixed message!

Solution:

ValueShuffle