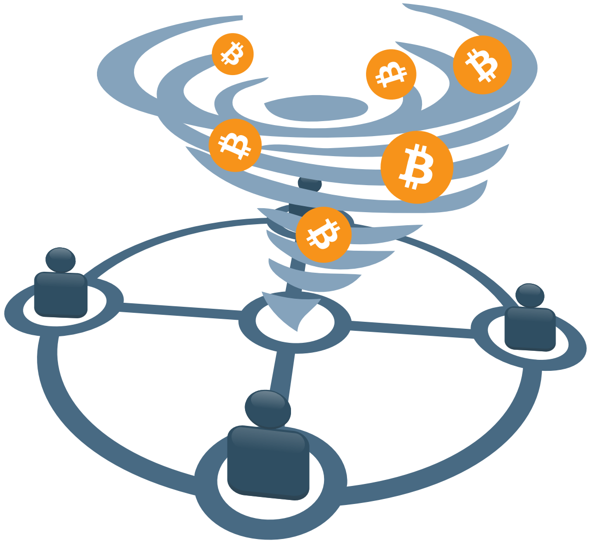


ValueShuffle: Mixing Confidential Transactions

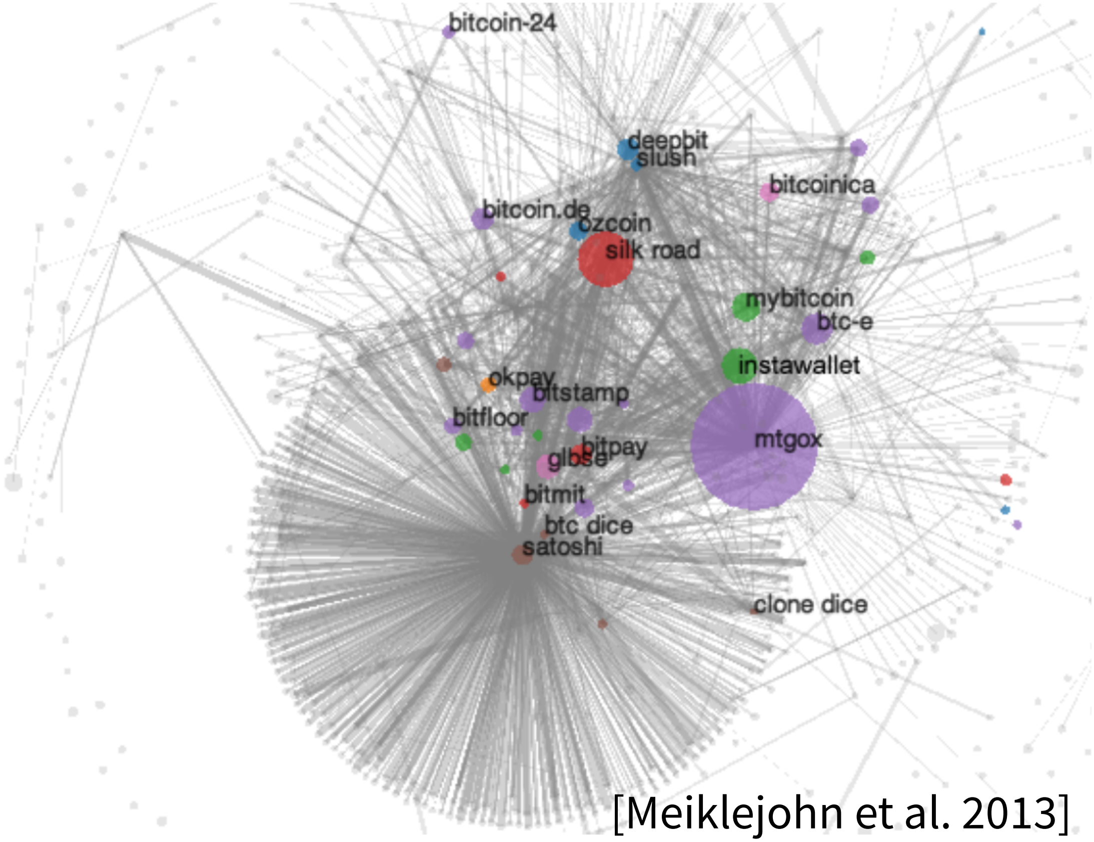
Tim Ruffing
@real_or_random

Pedro Moreno-Sanchez
@pedrorechez

UNIVERSITÄT DES SAARLANDES

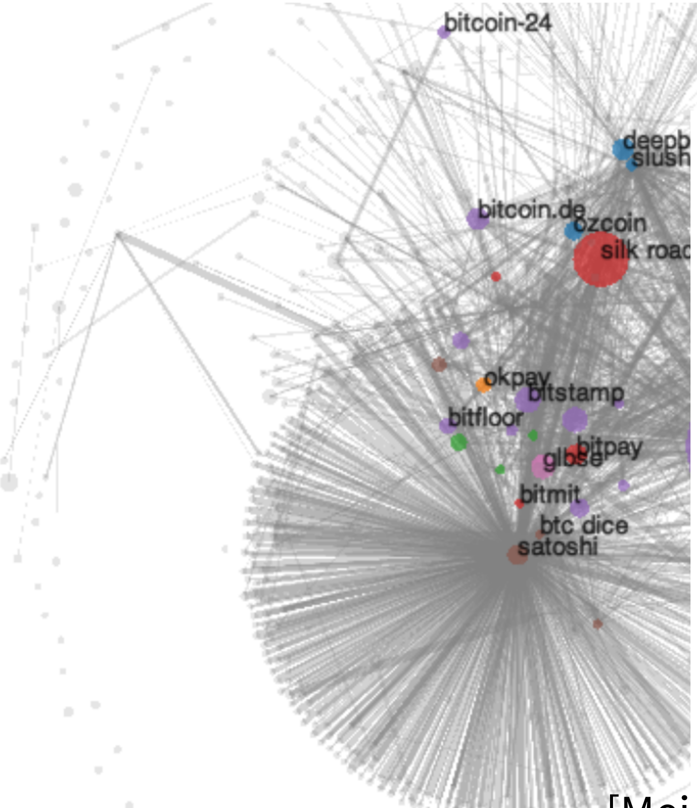
PURDUE UNIVERSITY

Linkability and Deanonymization Attacks

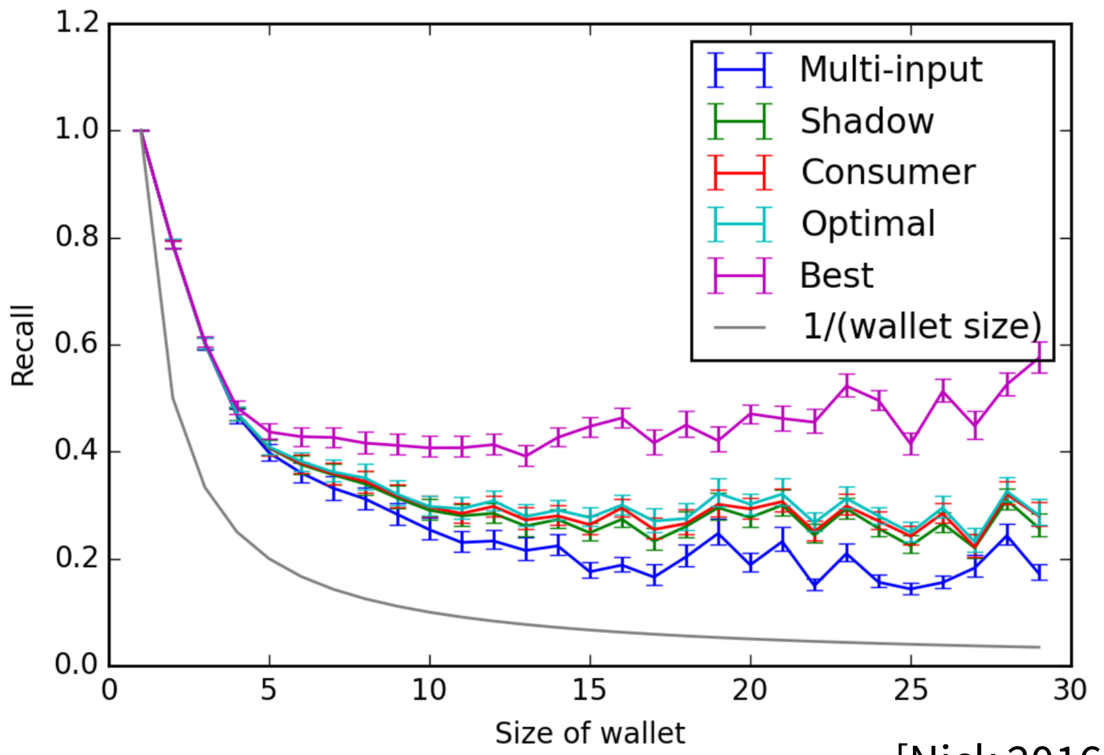


[Meiklejohn et al. 2013]

Linkability and Deanonymization Attacks

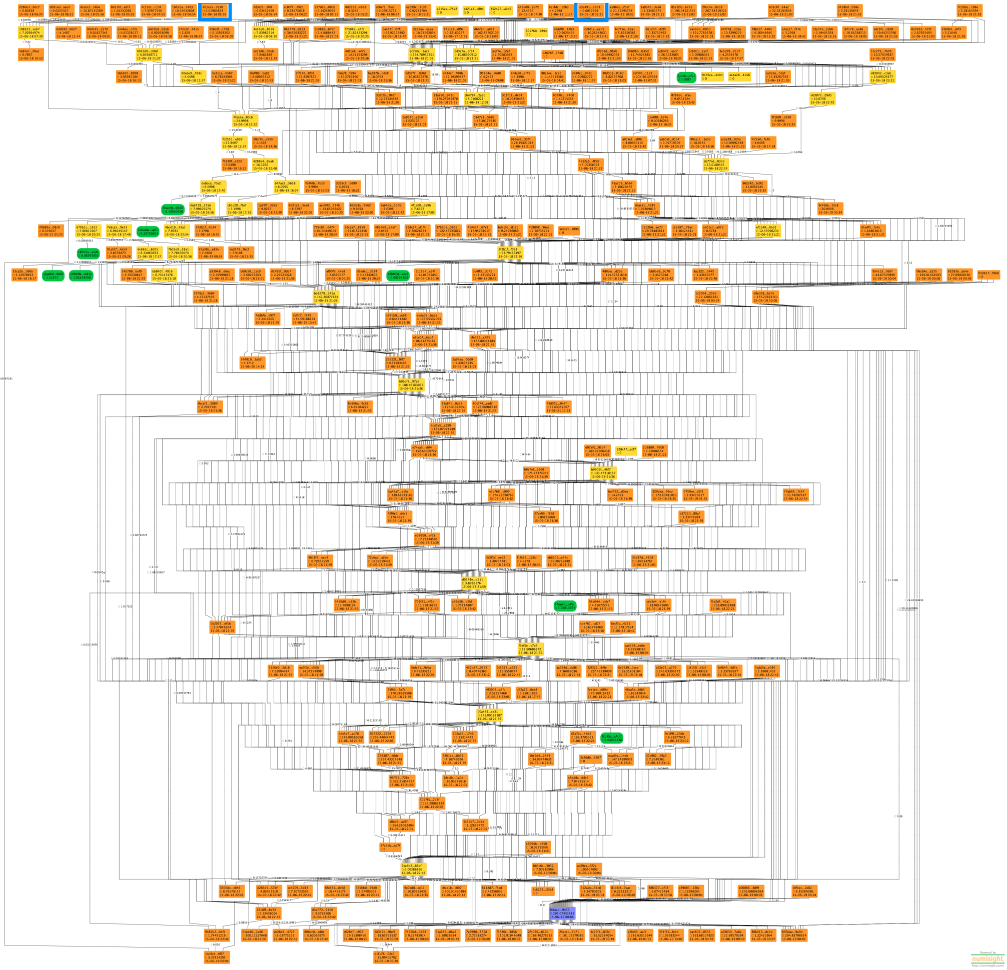


[Mei

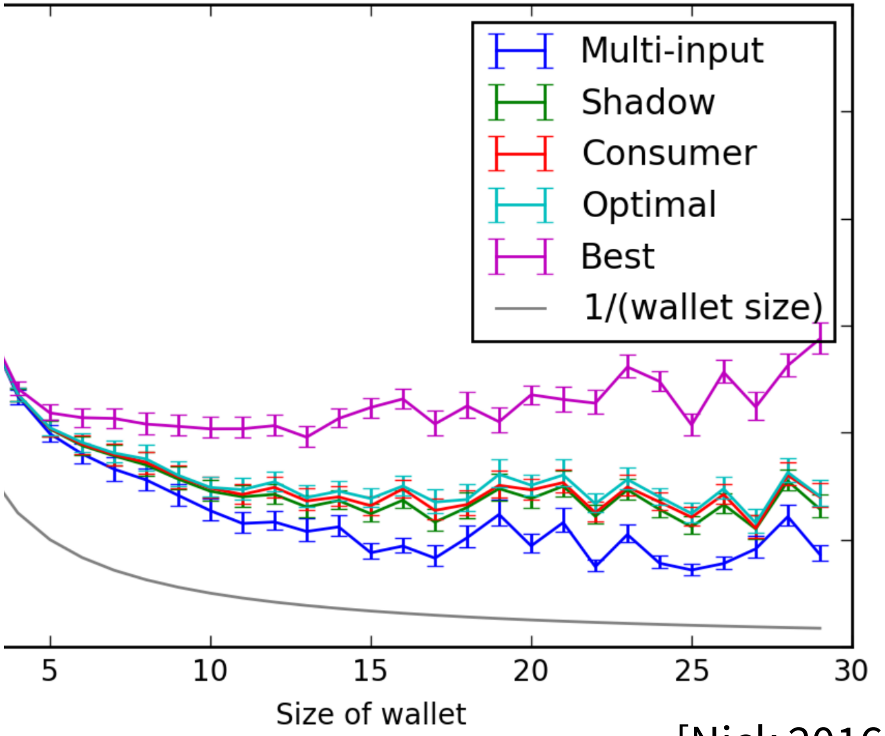


[Nick 2016]

Linkability and Deanonymization Attacks

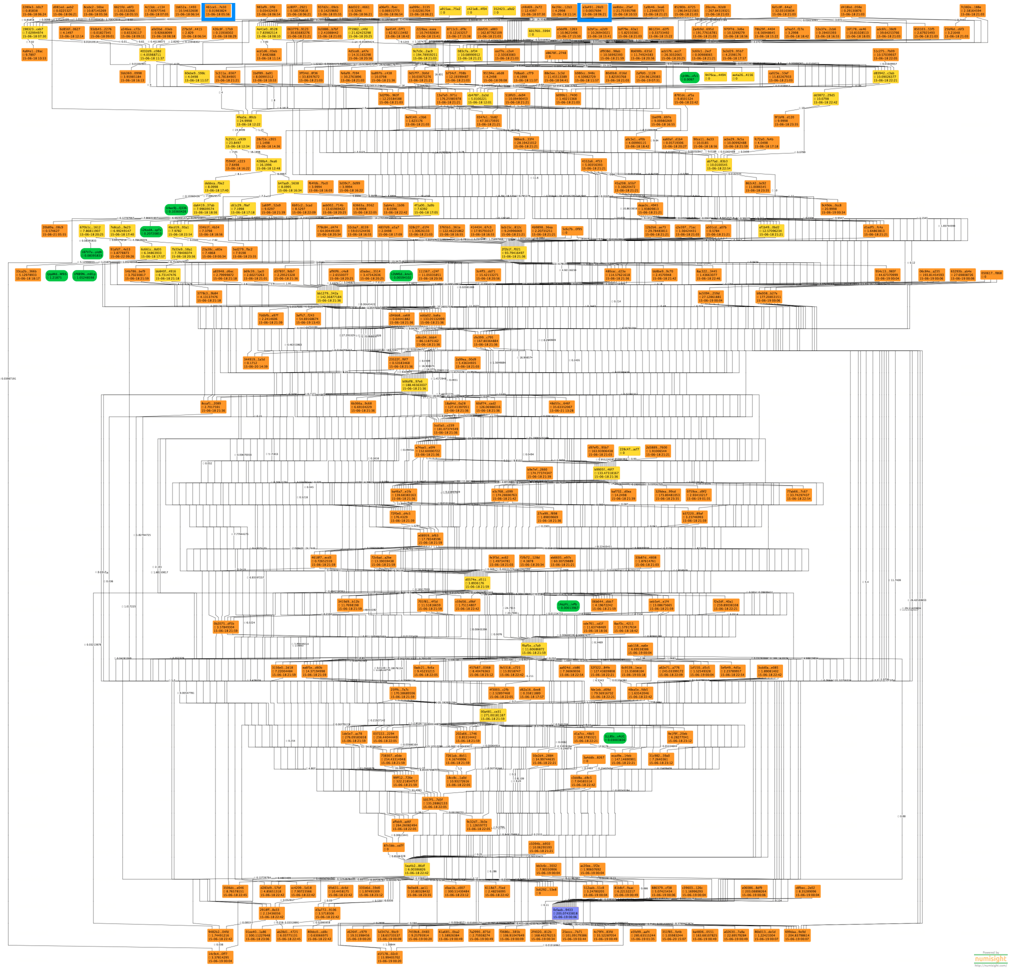


BitIodine [Spagnuolo, Maggi, Zanero 2013]

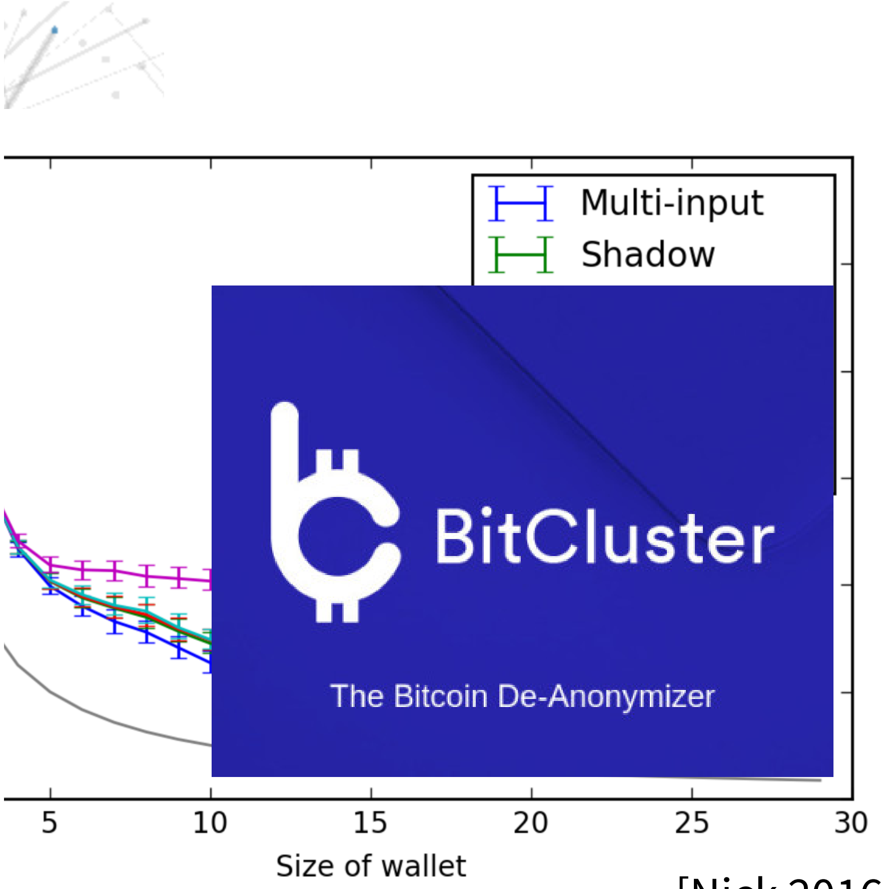


[Nick 2016]

Linkability and Deanonymization Attacks

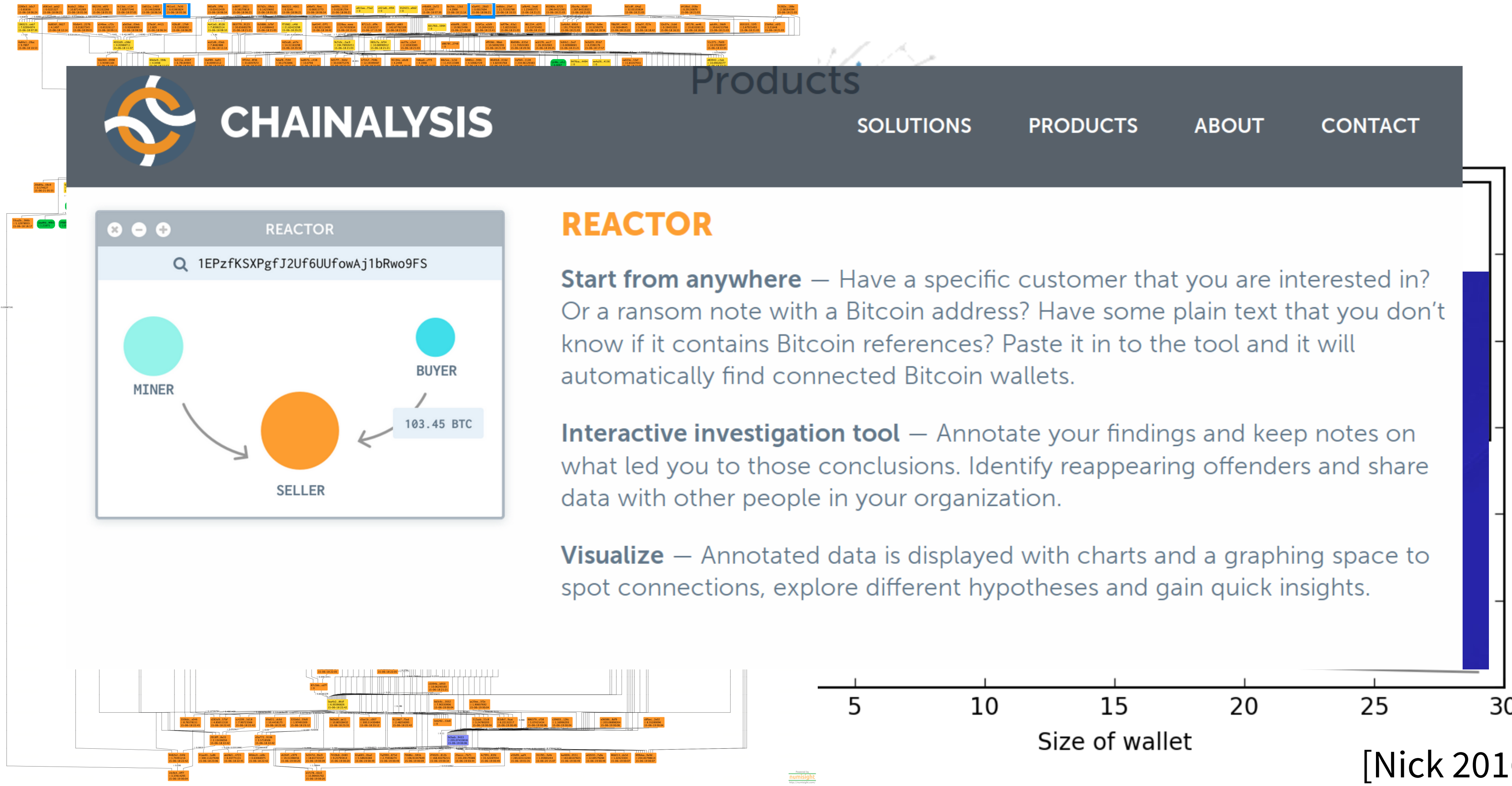


BitIodine [Spagnuolo, Maggi, Zanero 2013]



[Nick 2016]

Linkability and Deanonymization Attacks



Products

CHAINALYSIS

SOLUTIONS PRODUCTS ABOUT CONTACT

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.

[Nick 2016]

BitIodine [Spagnuolo, Maggi, Zanero 2013]

CoinJoin

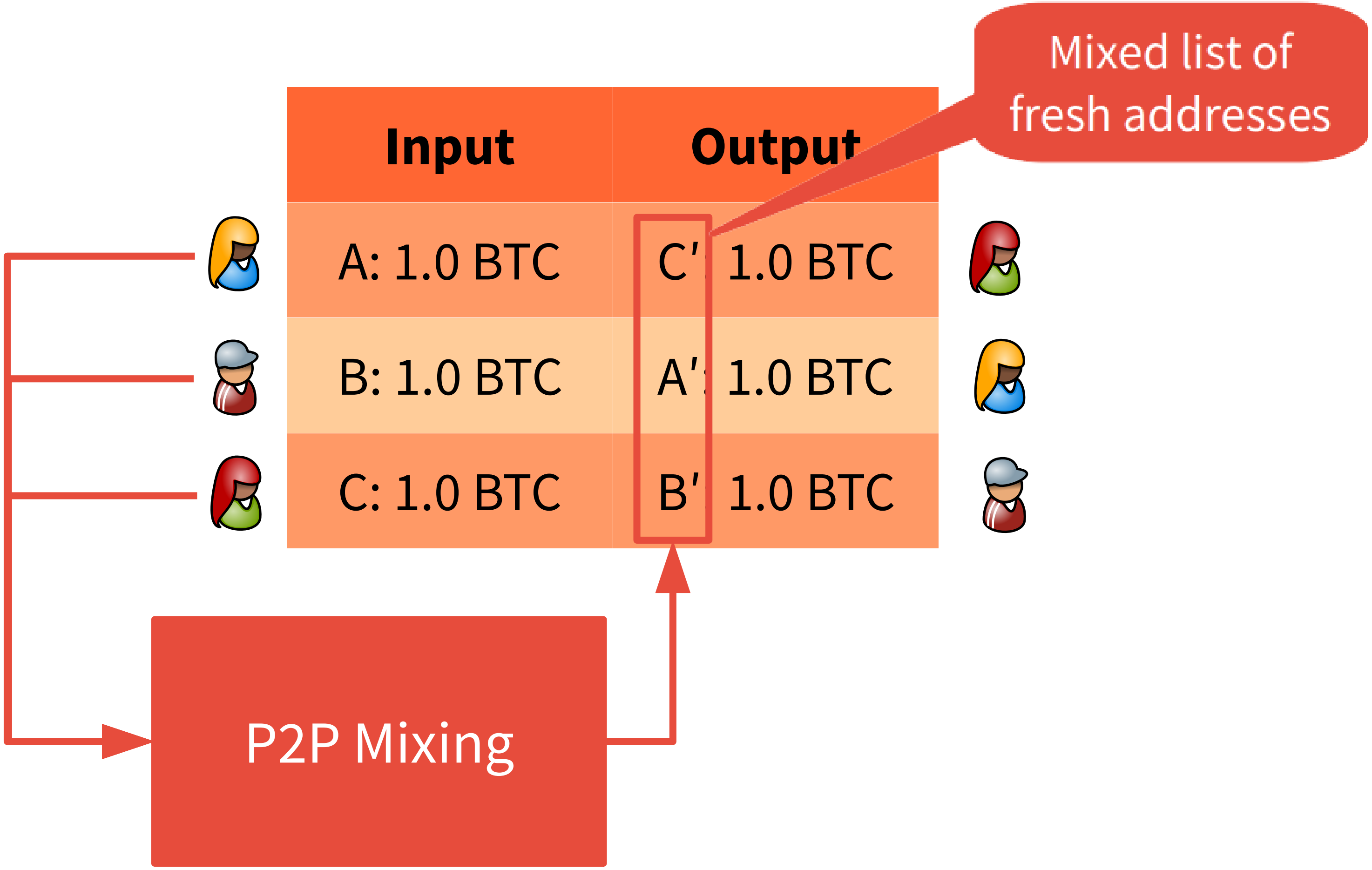
Input	Output
A: 1.0 BTC	C′: 1.0 BTC
B: 1.0 BTC	A′: 1.0 BTC
C: 1.0 BTC	B′: 1.0 BTC

CoinJoin

Input	Output
A: 1.0 BTC	C': 1.0 BTC
B: 1.0 BTC	A': 1.0 BTC
C: 1.0 BTC	B': 1.0 BTC

Mixed list of fresh addresses

CoinJoin



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

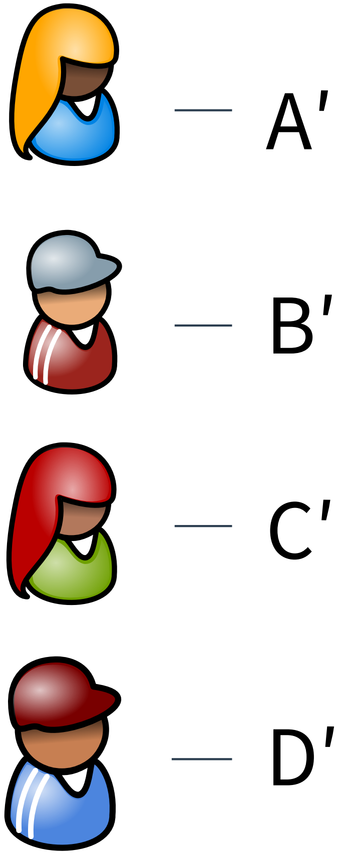
Mixed list of fresh addresses

P2P Mixing

DiceMix: An Efficient P2P Mixing Protocol

Tim Ruffing, Pedro Moreno-Sanchez, Aniket Kate. NDSS 2017

P2P Mixing



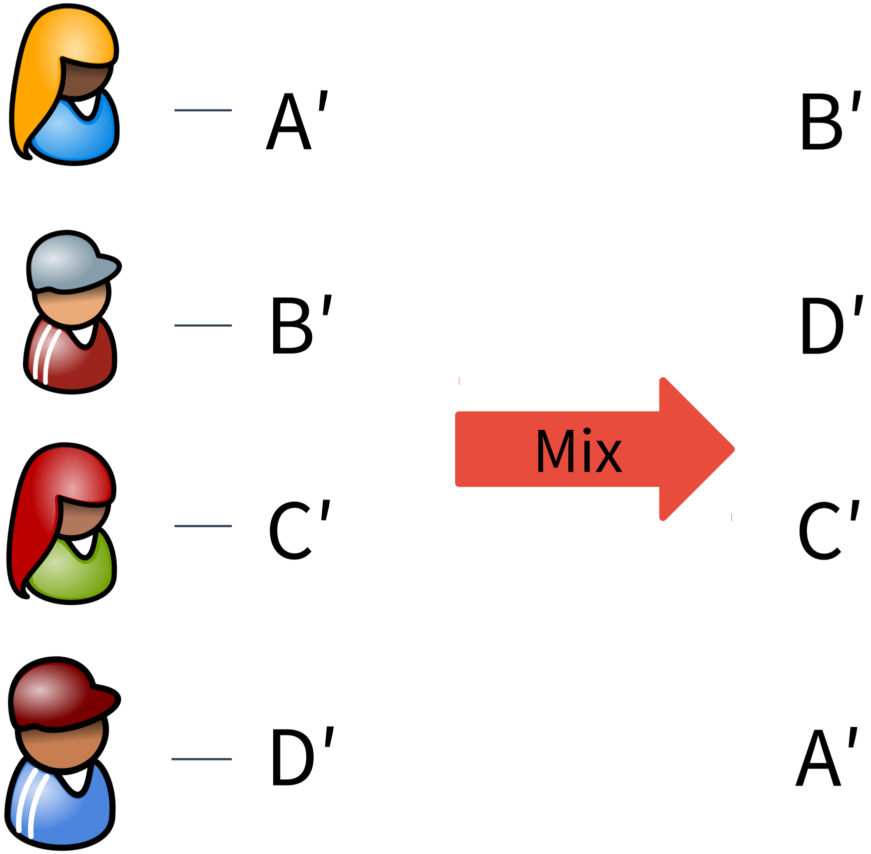
A'

B'

C'

D'

P2P Mixing



A'

B'

C'

D'

Mix

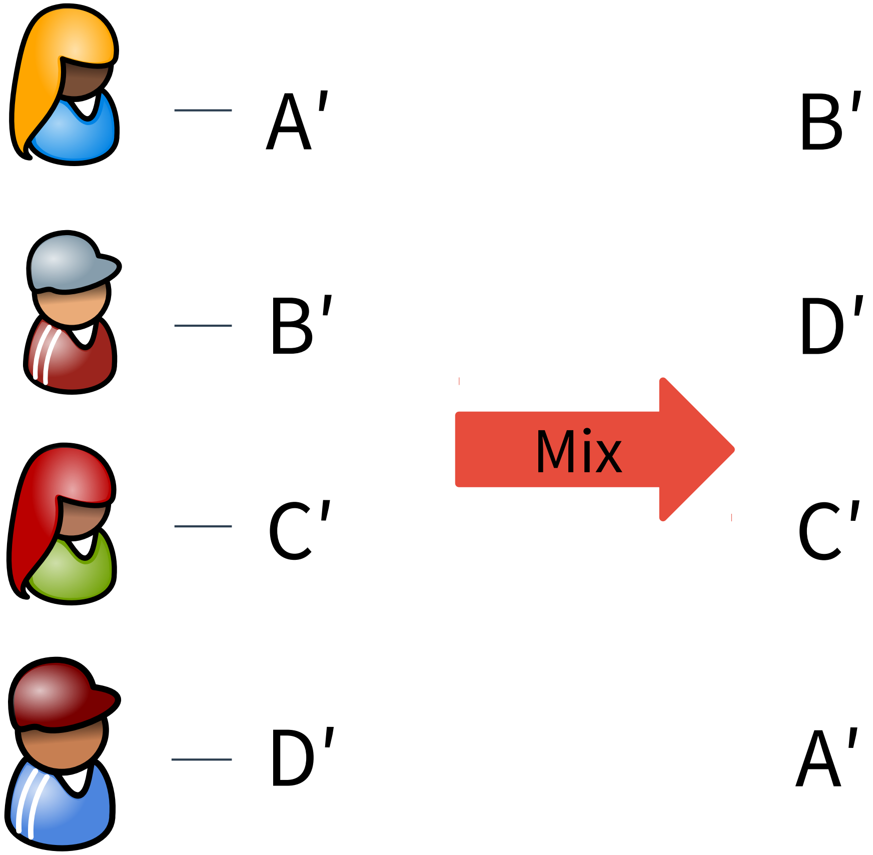
B'

D'

C'

A'

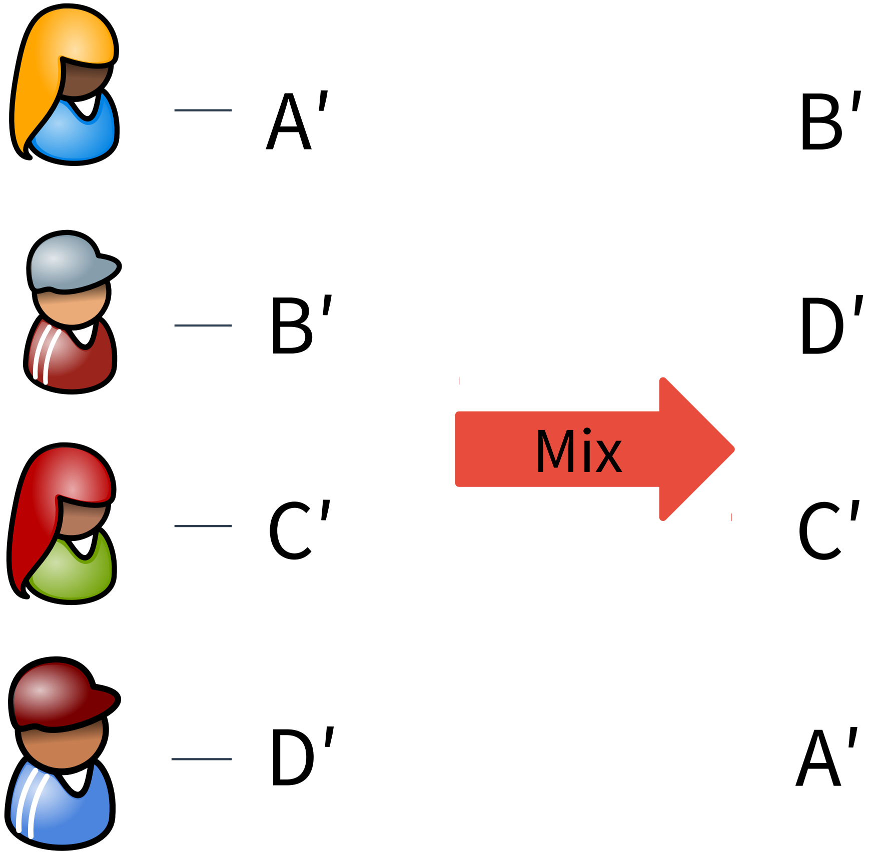
P2P Mixing



Confirmation

- Peers agree on the output and sign it

P2P Mixing



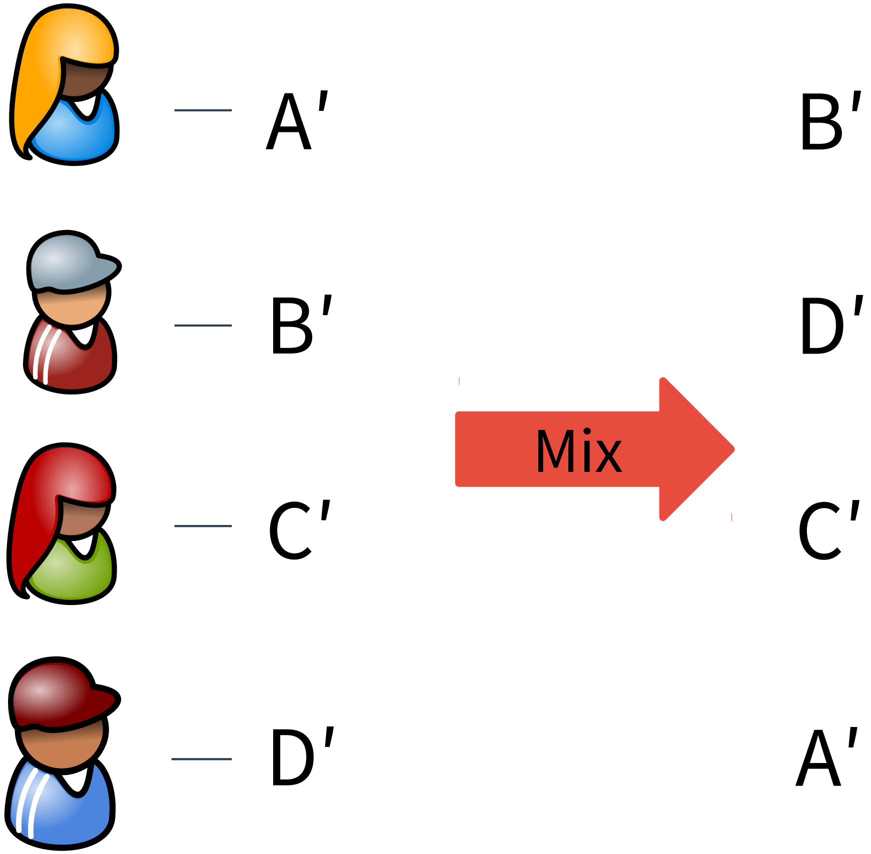
Confirmation

- Peers agree on the output and sign it

P2P Trust model

- No mutual trust, no third-party anonymity routers

P2P Mixing



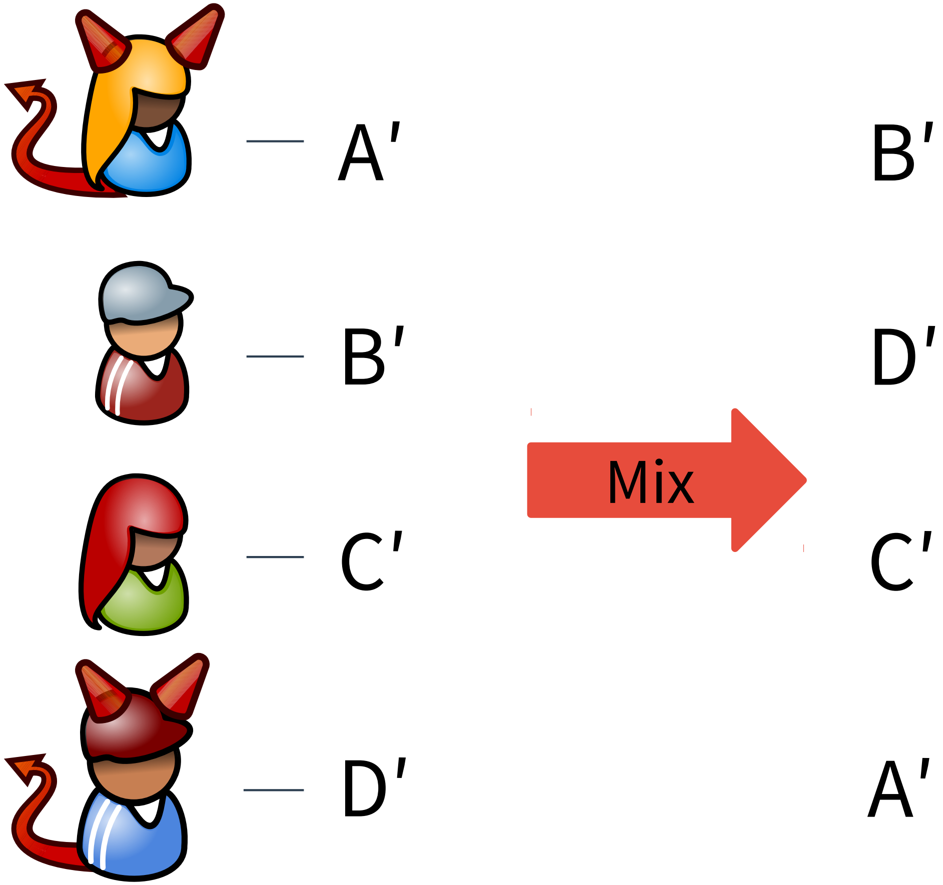
Confirmation

- Peers agree on the output and sign it

P2P Trust model

- No mutual trust, no third-party anonymity routers
- Bulletin board for communication, no trust

P2P Mixing



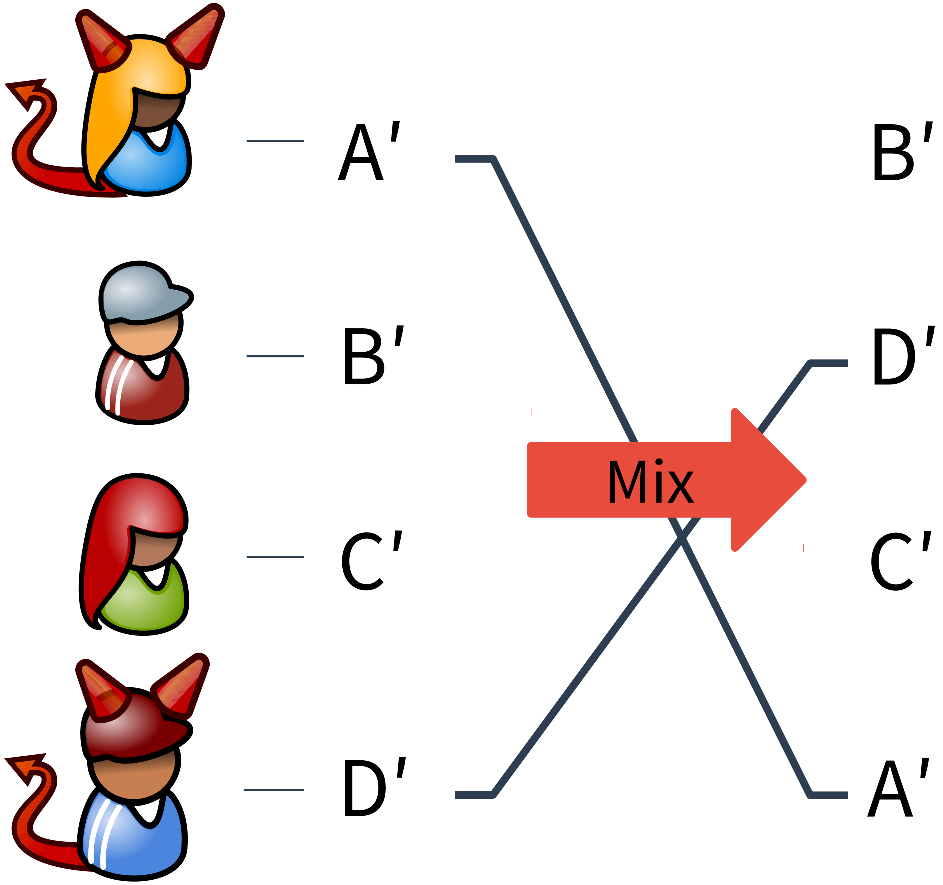
Confirmation

- Peers agree on the output and sign it

P2P Trust model

- No mutual trust, no third-party anonymity routers
- Bulletin board for communication, no trust

P2P Mixing



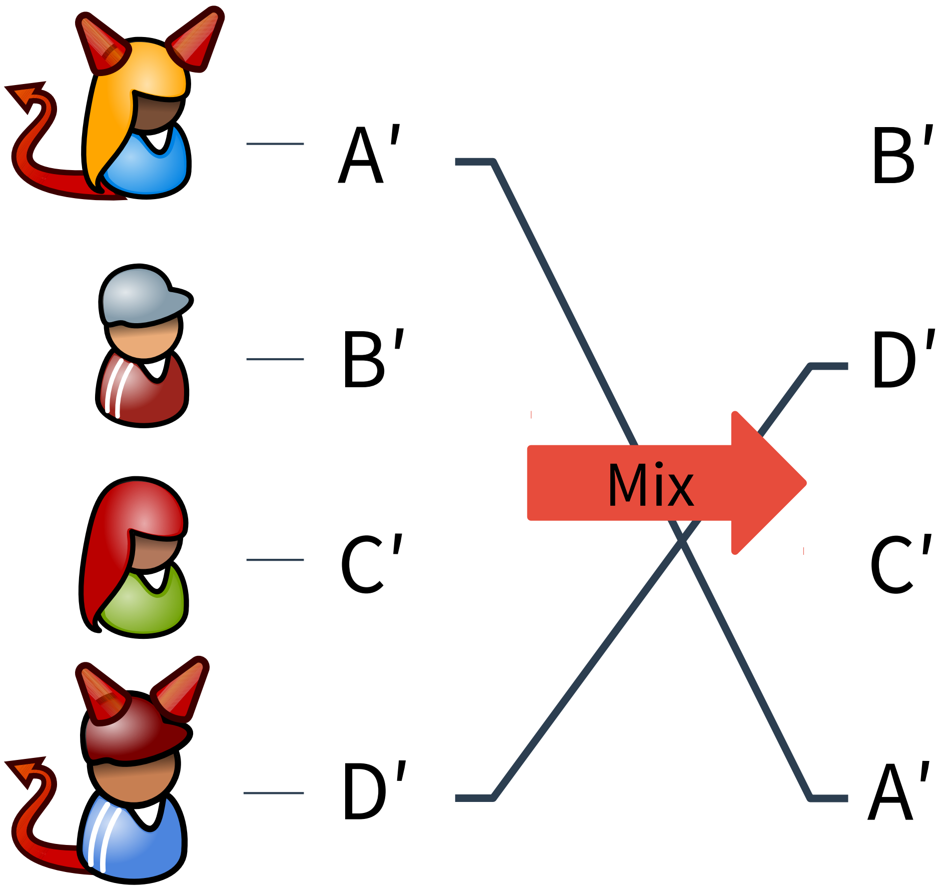
Confirmation

- Peers agree on the output and sign it

P2P Trust model

- No mutual trust, no third-party anonymity routers
- Bulletin board for communication, no trust
- Anoymity set is the set of honest users

P2P Mixing



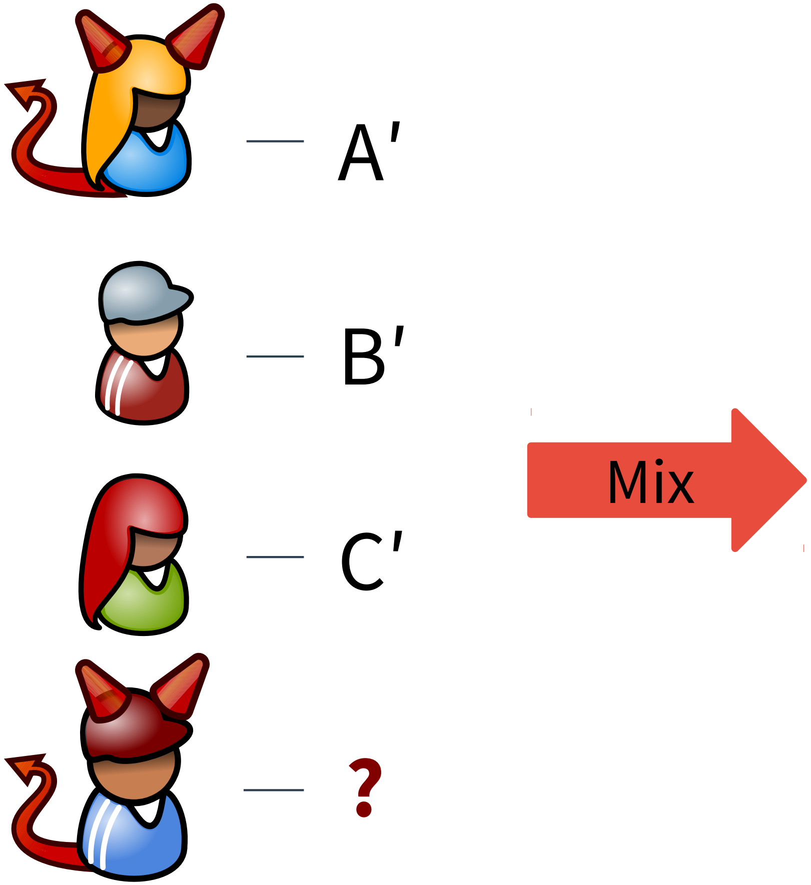
Confirmation

- Peers agree on the output and sign it

P2P Trust model

- No mutual trust, no third-party anonymity routers
- Bulletin board for communication, no trust
- Anoymity set is the set of honest users
- Protocol must terminate in the presence of malicious users

Disruption



— A′

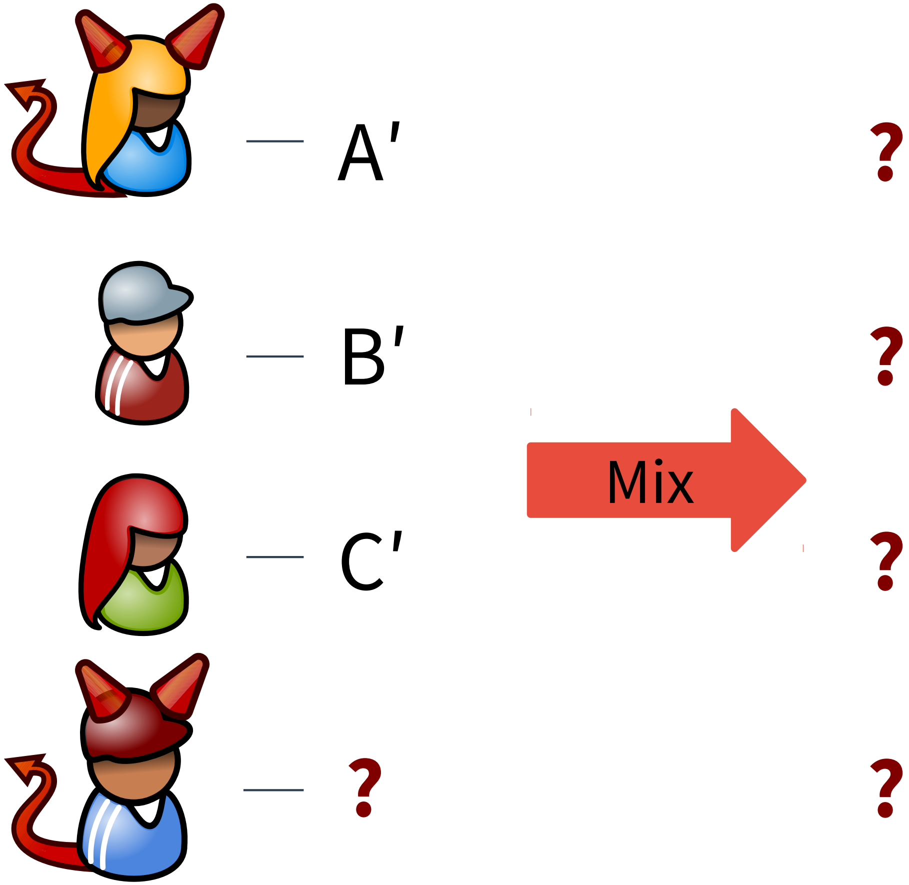
— B′

— C′

— ?

Mix

Disruption



A′ ?

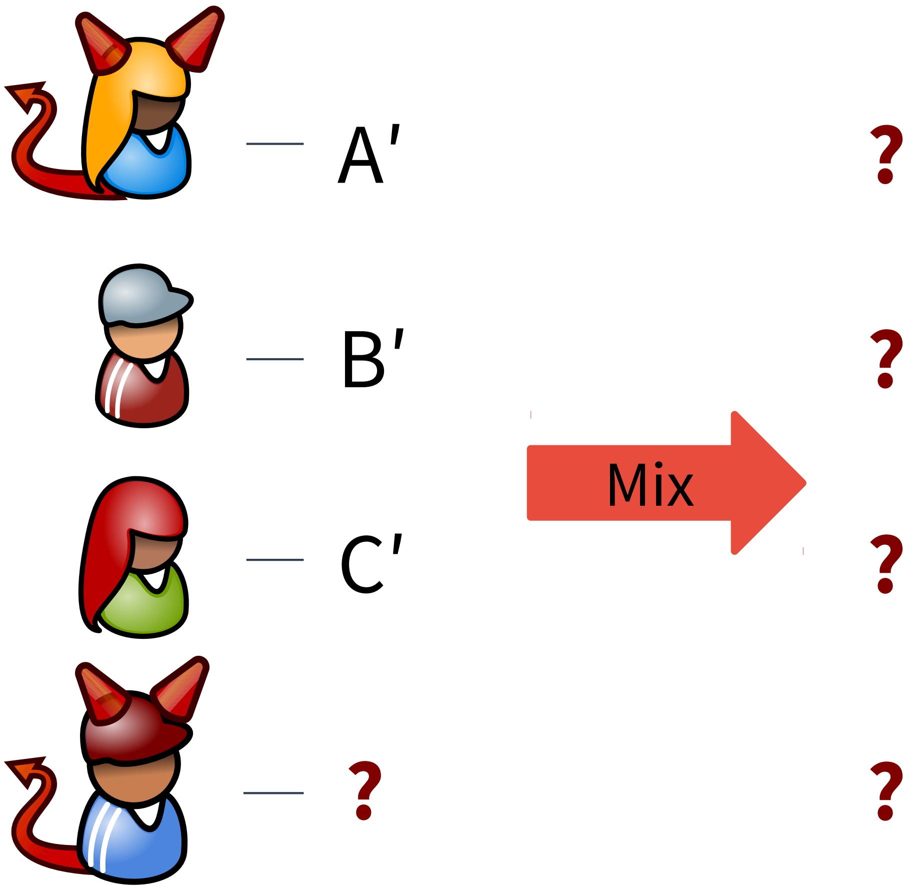
B′ ?

Mix

C′ ?

? ?

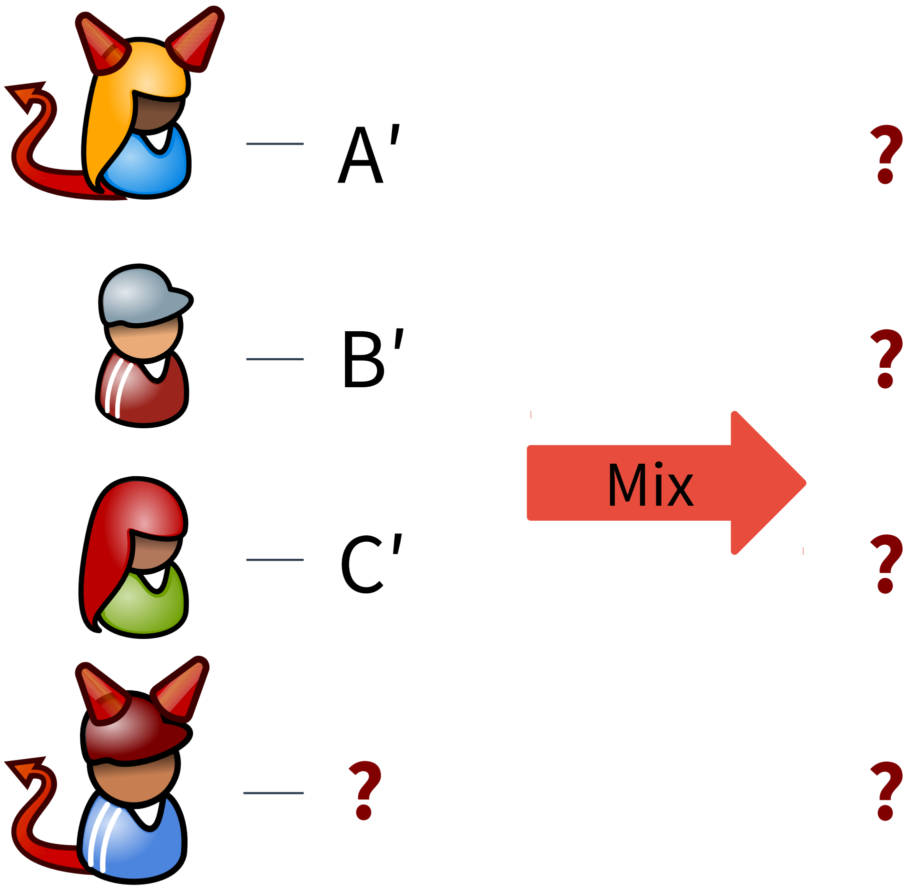
Disruption



Goal:

Kick out the disrupting user and start from scratch.

Disruption



Goal:

Kick out the disrupting user and start from scratch.

Problem:

Anonymity

Handling Disruptions

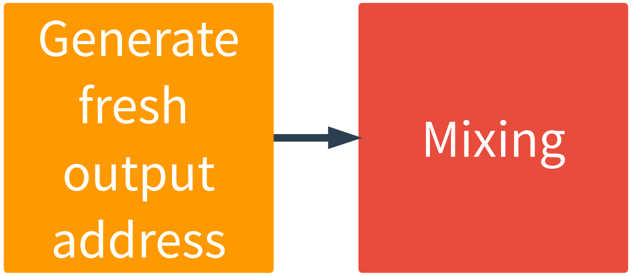
Handling Disruptions

IN CASE OF
DISRUPTION
BREAK ANONYMITY

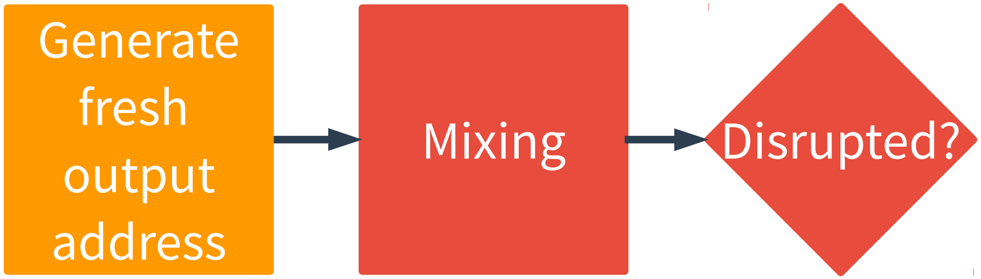
Flowchart of DiceMix + CoinJoin (= CoinShuffle++)

Generate fresh output address

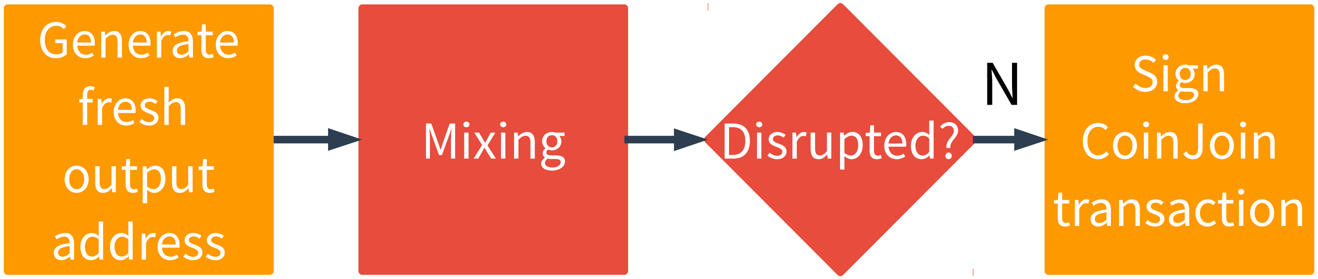
Flowchart of DiceMix + CoinJoin (= CoinShuffle++)



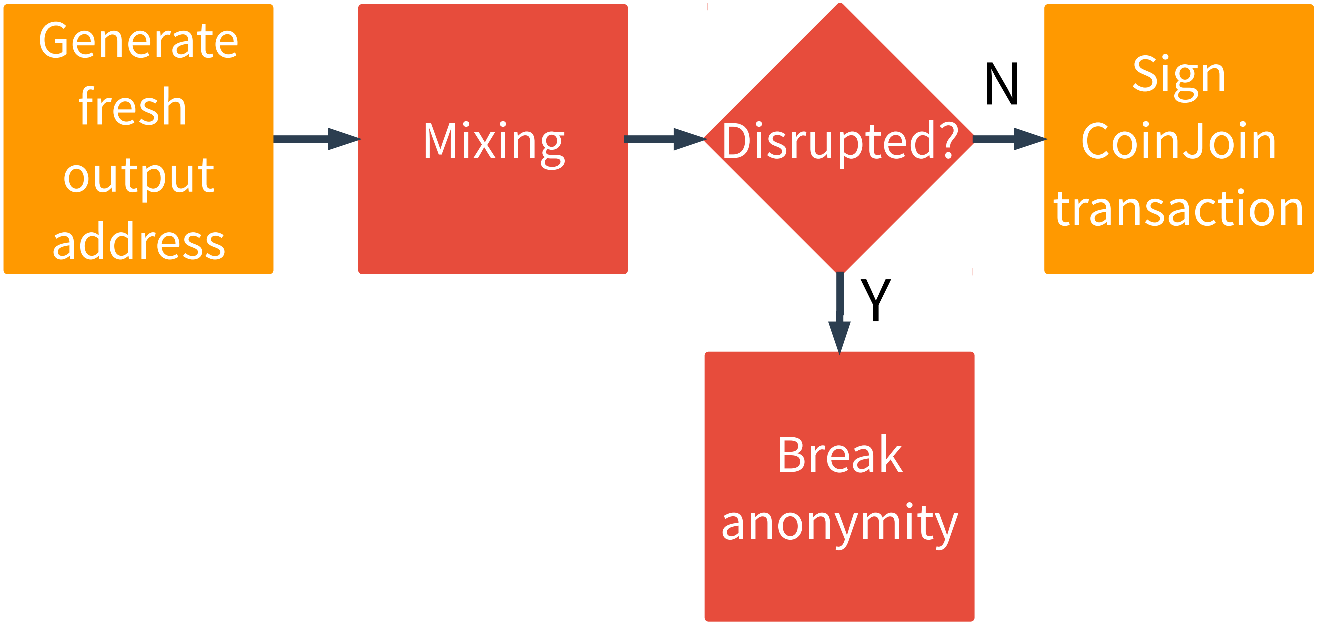
Flowchart of DiceMix + CoinJoin (= CoinShuffle++)



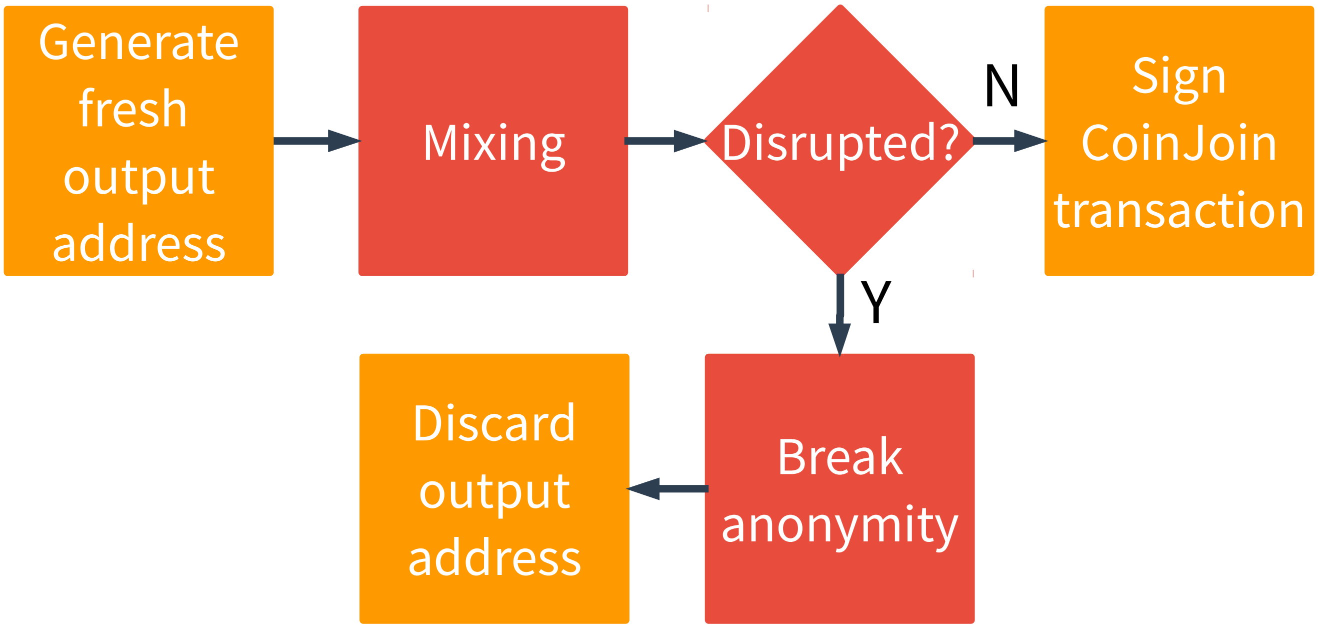
Flowchart of DiceMix + CoinJoin (= CoinShuffle++)



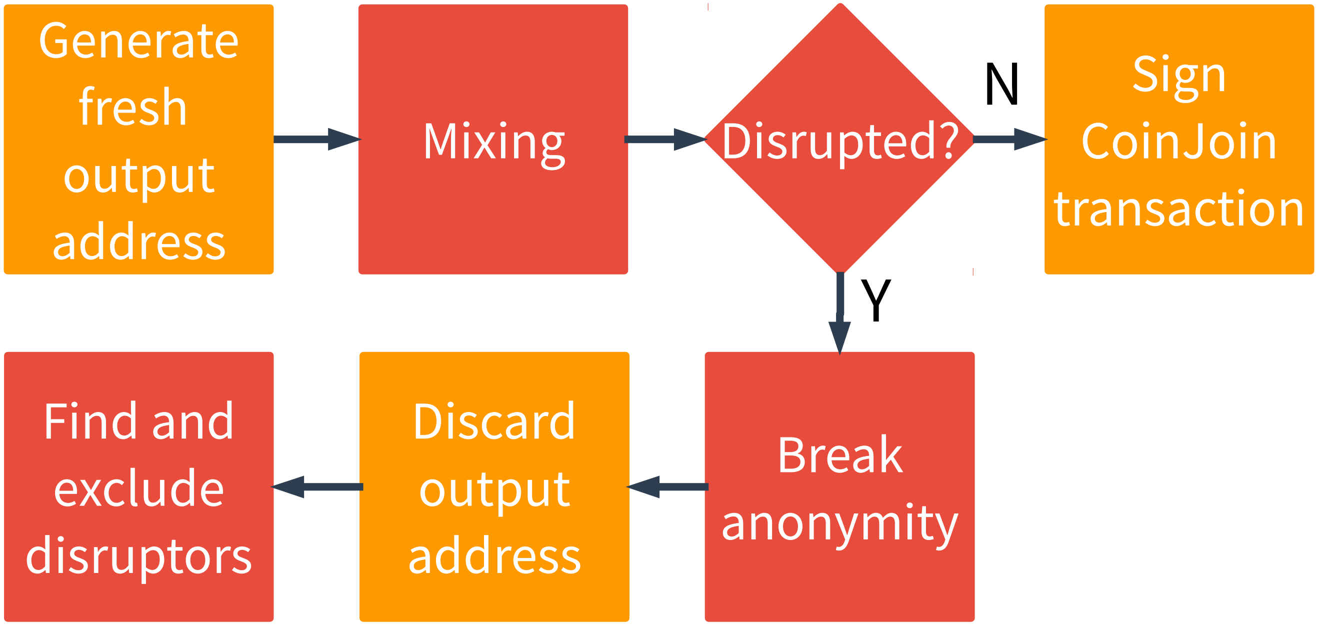
Flowchart of DiceMix + CoinJoin (= CoinShuffle++)



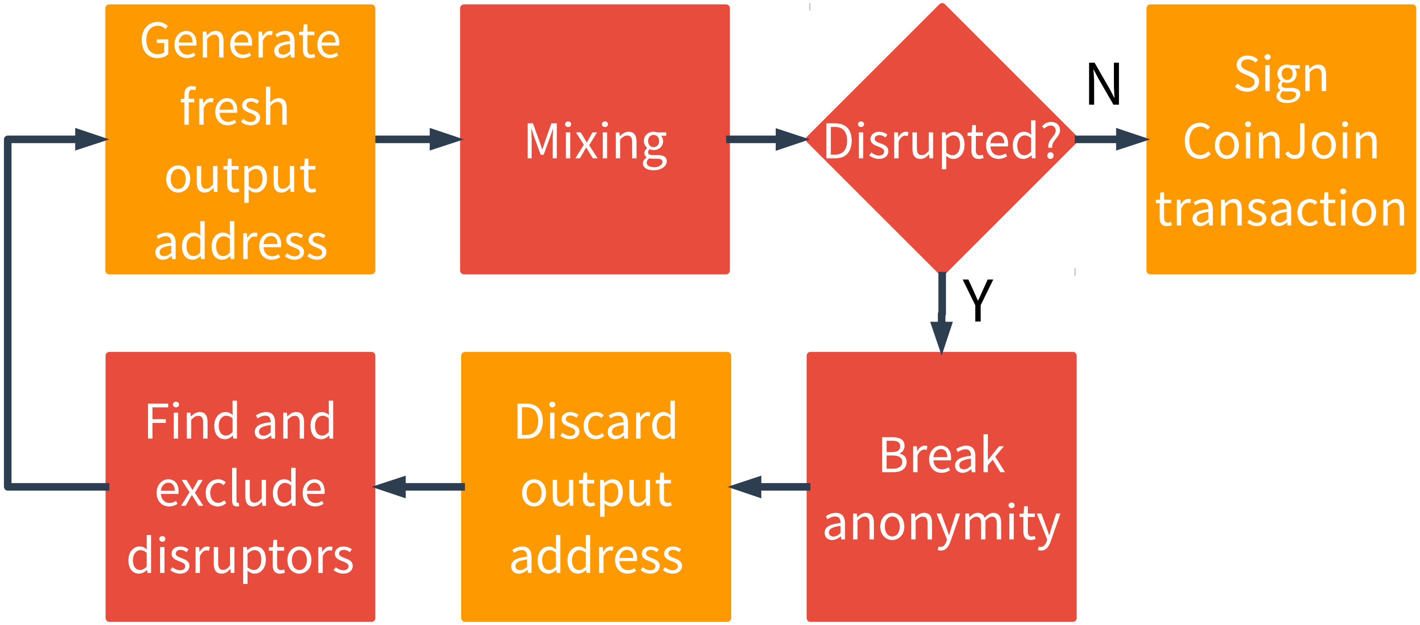
Flowchart of DiceMix + CoinJoin (= CoinShuffle++)



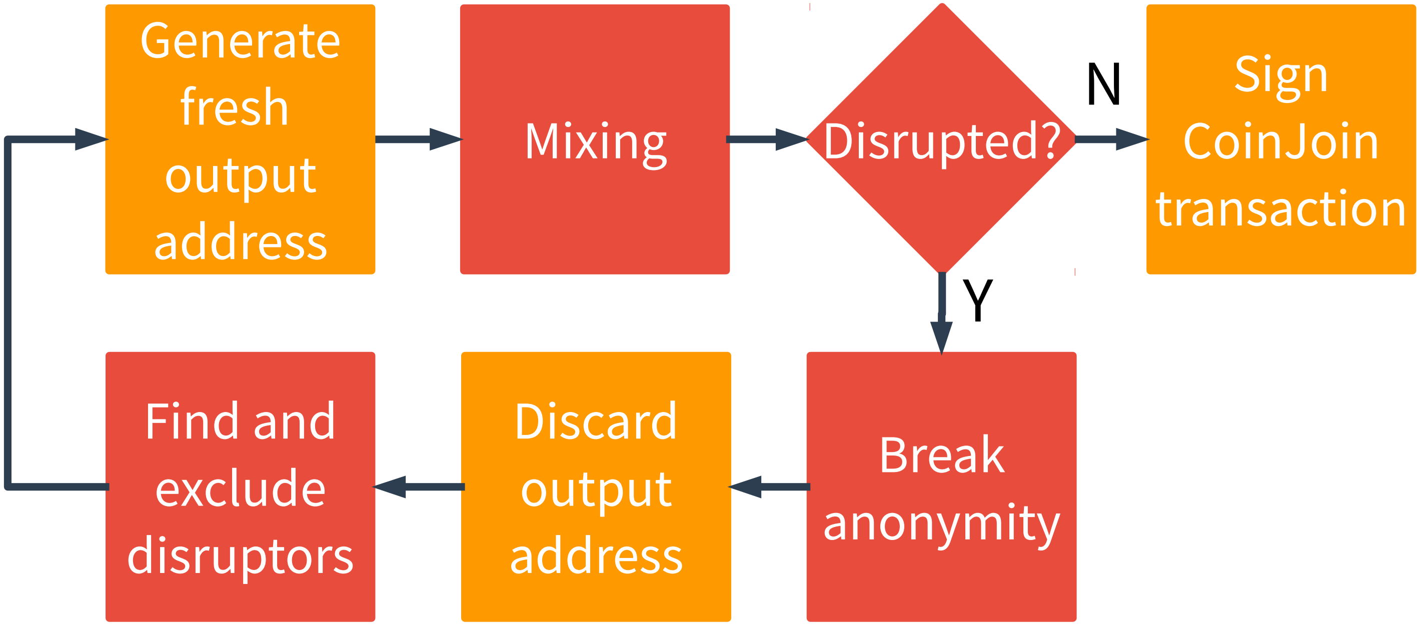
Flowchart of DiceMix + CoinJoin (= CoinShuffle++)


Generate fresh output address
Mixing
Disrupted?
N
Sign CoinJoin transaction
Y
Break anonymity
Discard output address
Find and exclude disruptors

Flowchart of DiceMix + CoinJoin (= CoinShuffle++)

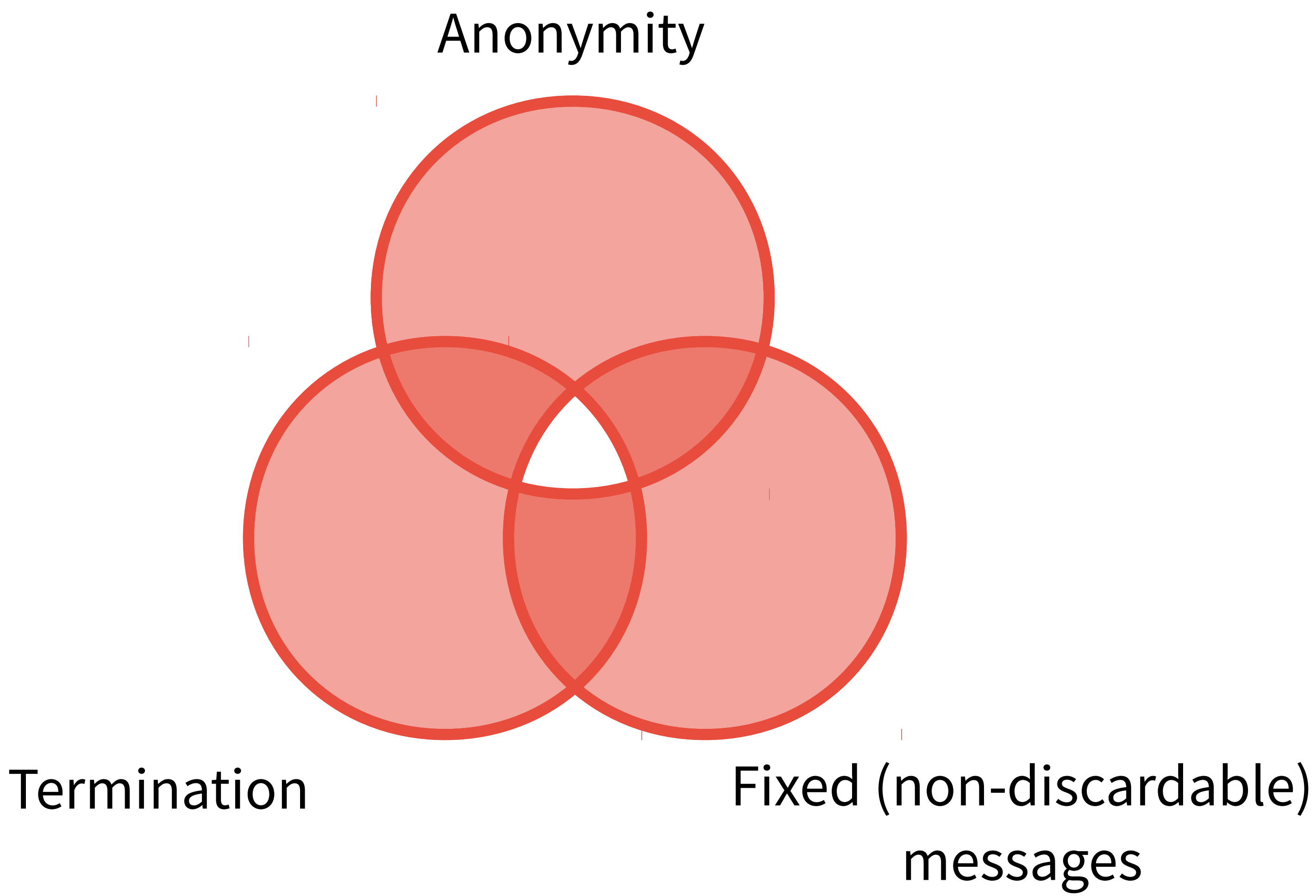

Generate fresh output address
Mixing
Disrupted?
N
Sign CoinJoin transaction
Y
Break anonymity
Discard output address
Find and exclude disruptors

Flowchart of DiceMix + CoinJoin (= CoinShuffle++)

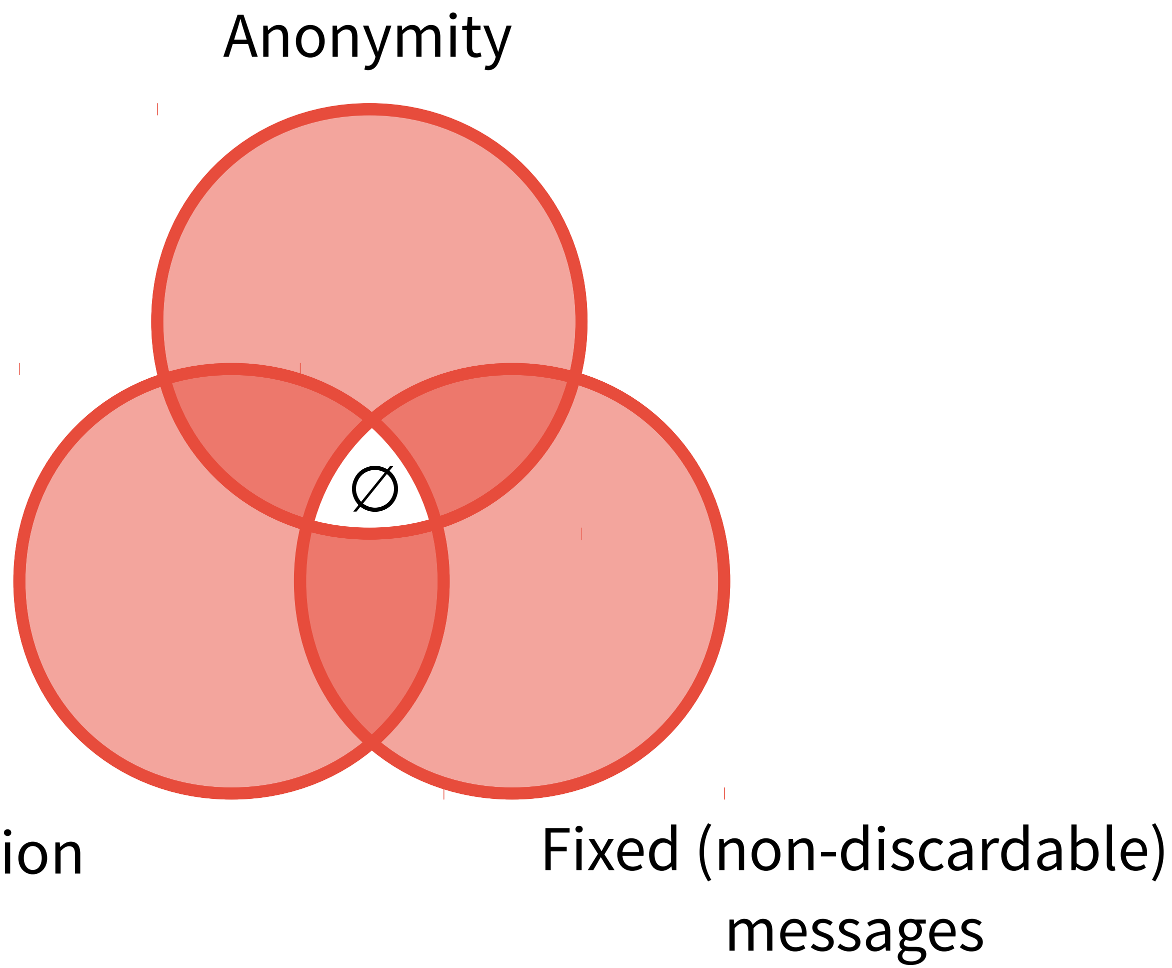

Generate fresh output address
Mixing
Disrupted?
N
Sign CoinJoin transaction
Y
Break anonymity
Discard output address
Find and exclude disruptors

Possible because addresses are discardable

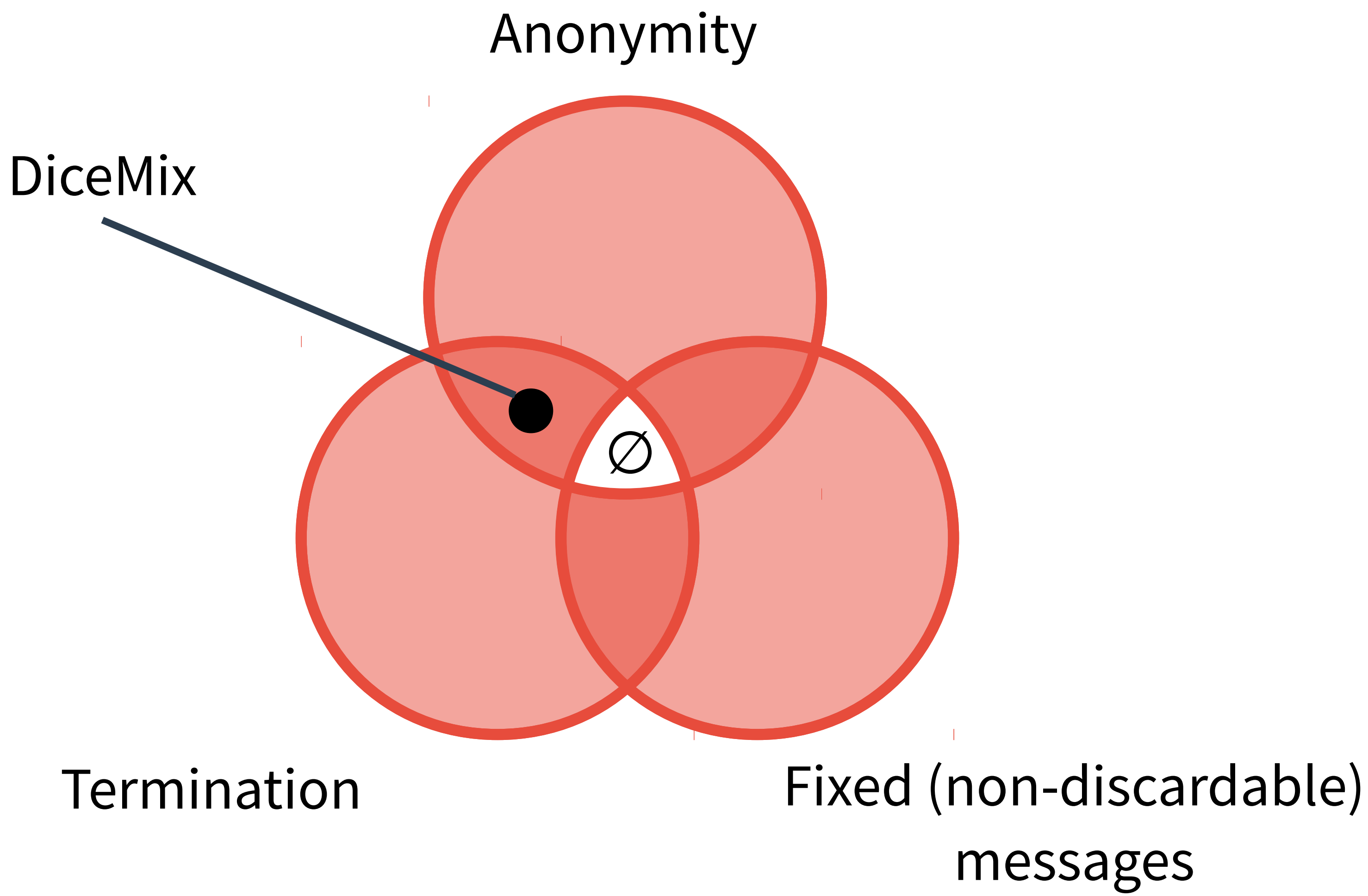
Discardability in P2P Mixing



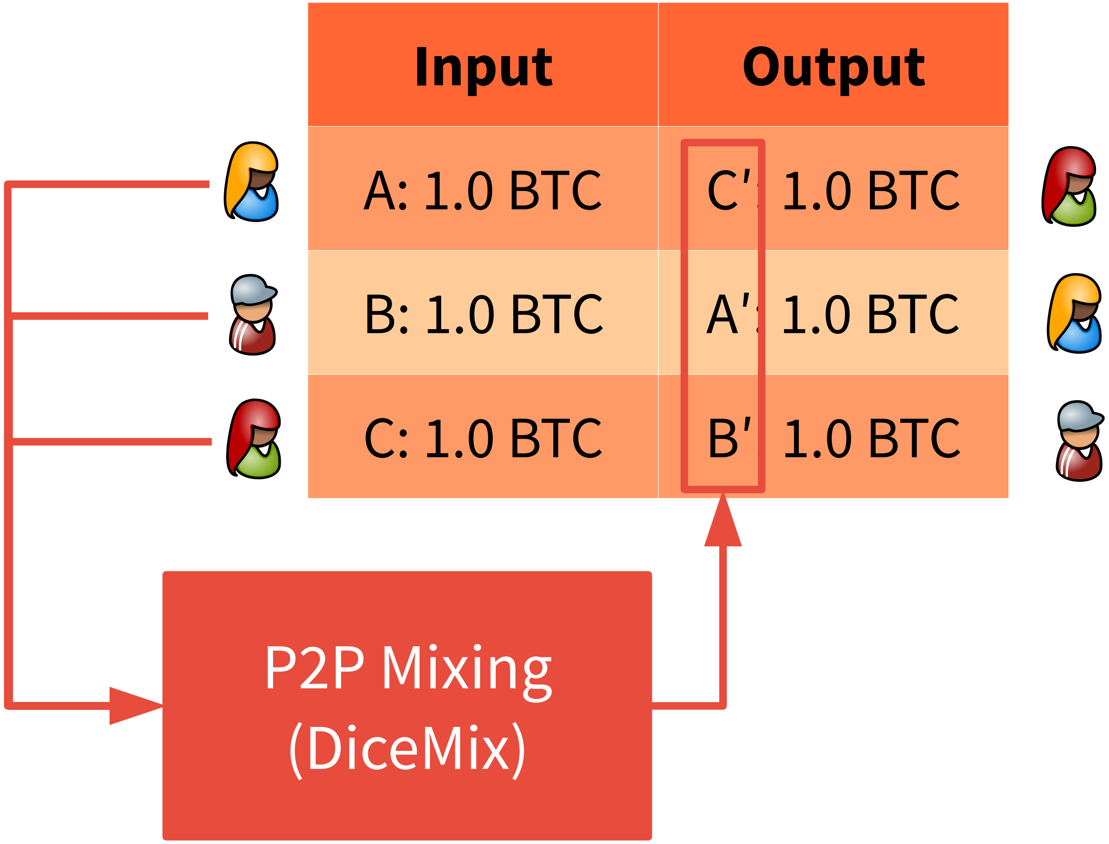
Discardability in P2P Mixing



Discardability in P2P Mixing



Mixing



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

P2P Mixing
(DiceMix)

Why Mixing Sucks: A Play in Three Acts

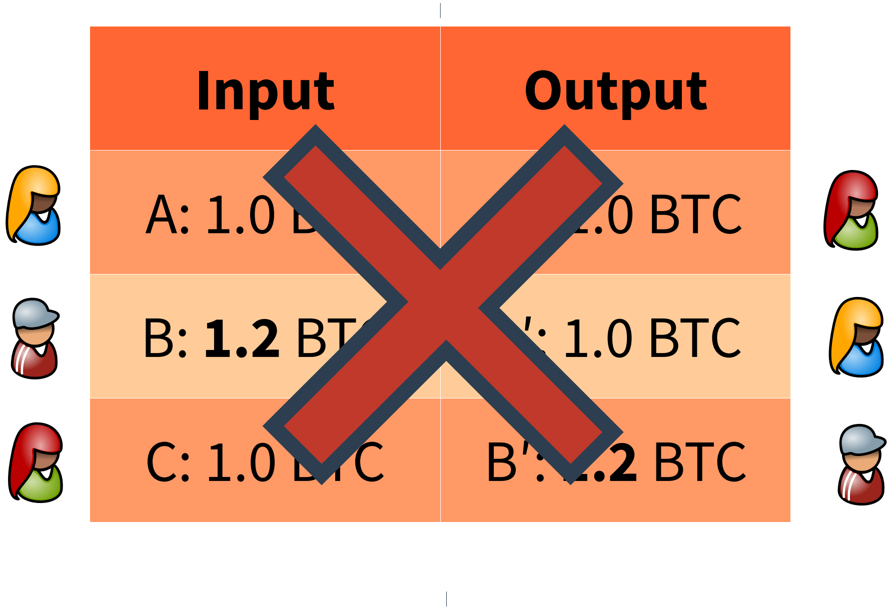
Bob wants to Mix Coins

WANTS TO MIX
COINS
memegenerator.net

1. Mixing Sucks

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. Mixing Sucks



Input	Output
A: 1.0 B	.0 BTC
B: **1.2** BT	': 1.0 BTC
C: 1.0 B TC	B': **.2** BTC

1. Mixing Sucks

Input	Output
A: 1.0 BTC	C′: 1.0 BTC
B: **1.2** BTC	A′: 1.0 BTC
C: 1.0 BTC	B′: **1.0** BTC
	B″: **0.2** BTC

1. Mixing Sucks

Input	Output
A: 1.0 BTC	C′: 1.0 BTC
B: **1.2** BTC	A′: 1.0 BTC
C: 1.0 BTC	B′: **1.0** BTC
	B′′: **0.2** BTC

What to do
with the change?

2. Mixing Sucks Even More

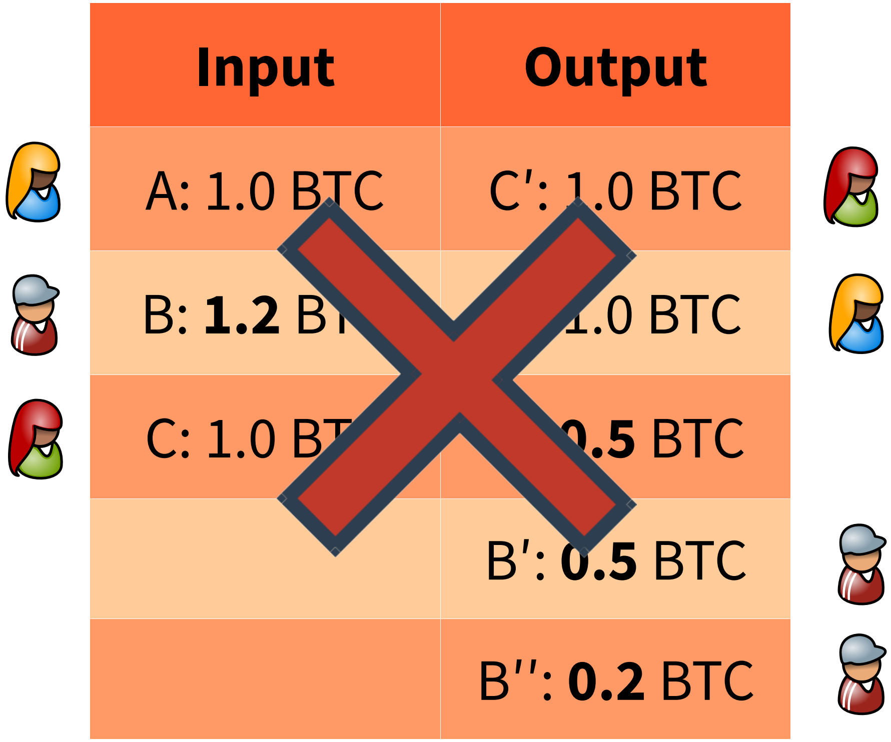
Input	Output
A: 1.0 BTC	C′: 1.0 BTC
B: **1.2** BTC	A′: 1.0 BTC
C: 1.0 BTC	R: **0.5** BTC
	B′: **0.5** BTC
	B″: **0.2** BTC

2. Mixing Sucks Even More

Input	Output
A: 1.0 BTC	C′: 1.0 BTC
B: **1.2** BTC	A′: 1.0 BTC
C: 1.0 BTC	R: **0.5** BTC
	B′: **0.5** BTC
	B′′: **0.2** BTC

Bob’s message in P2P mixing protocol:
(B′, 0.5)

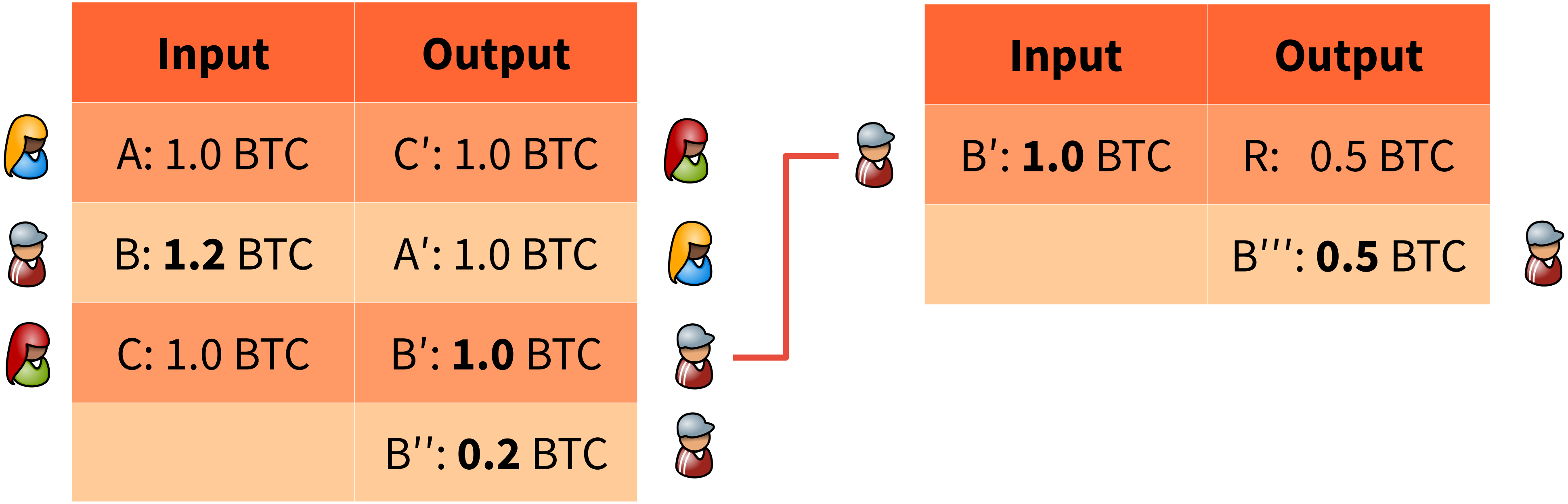
2. Mixing Sucks Even More



Input	Output
A: 1.0 BTC	C′: 1.0 BTC
B: **1.2** BTC	1.0 BTC
C: 1.0 BTC	**0.5** BTC
	B′: **0.5** BTC
	B″: **0.2** BTC

Bob’s message in P2P mixing protocol:
(B′, 0.5)

2. Mixing Sucks Even More



Input	Output
A: 1.0 BTC	C′: 1.0 BTC
B: **1.2** BTC	A′: 1.0 BTC
C: 1.0 BTC	B′: **1.0** BTC
	B′′: **0.2** BTC

Input	Output
B′: **1.0** BTC	R: 0.5 BTC
	B′′′: **0.5** BTC

2. Mixing Sucks Even More

Input	Output
A: 1.0 BTC	C′: 1.0 BTC
B: **1.2** BTC	A′: 1.0 BTC
C: 1.0 BTC	B′: **1.0** BTC
	B′′: **0.2** BTC

Input	Output
B′: **1.0** BTC	R: 0.5 BTC
	B′′′: **0.5** BTC

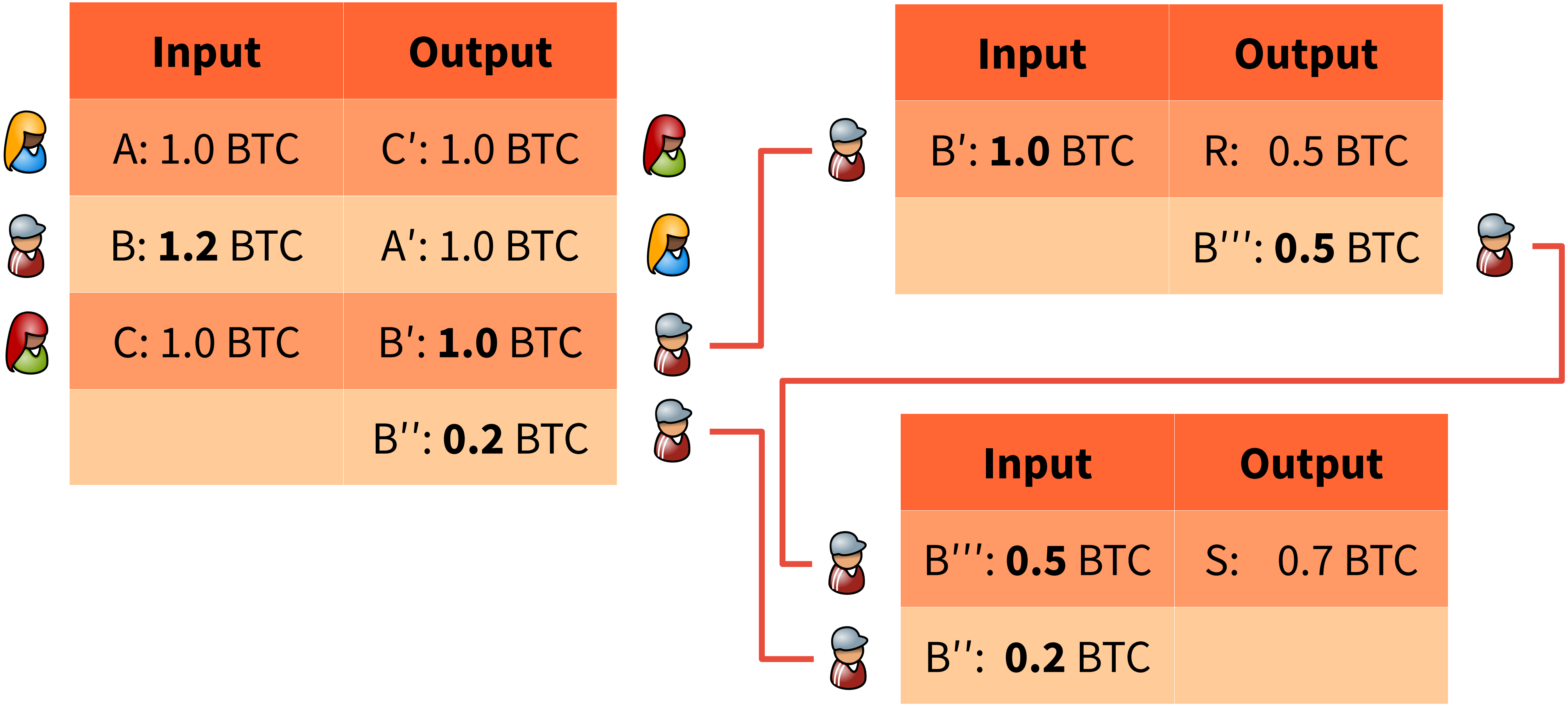
I need two transactions?!

3. Mixing Sucks, Really!

Input	Output
A: 1.0 BTC	C′: 1.0 BTC
B: **1.2** BTC	A′: 1.0 BTC
C: 1.0 BTC	B′: **1.0** BTC
	B″: **0.2** BTC

Input	Output
B′: **1.0** BTC	R: 0.5 BTC
	B‴: **0.5** BTC

3. Mixing Sucks, Really!

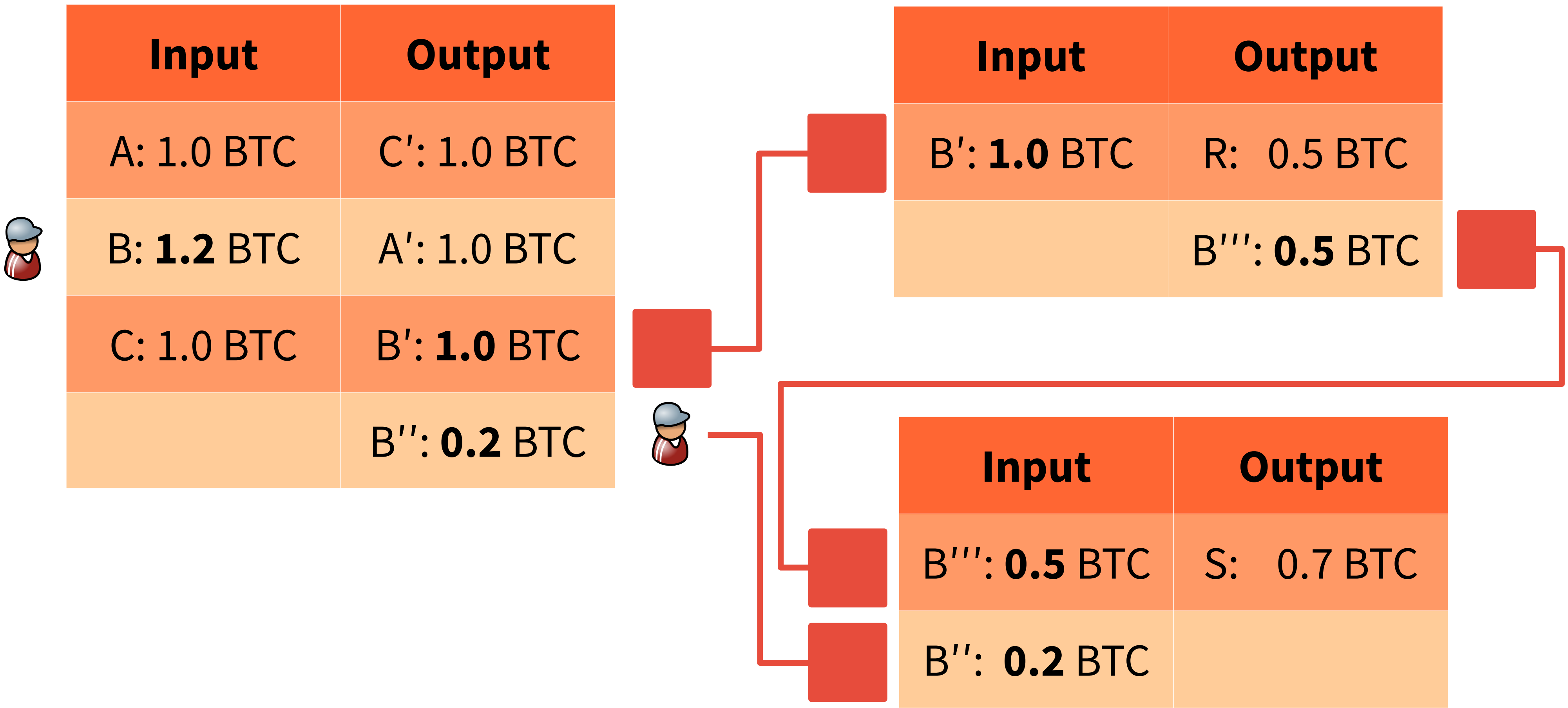


Input	Output
A: 1.0 BTC	C′: 1.0 BTC
B: **1.2** BTC	A′: 1.0 BTC
C: 1.0 BTC	B′: **1.0** BTC
	B″: **0.2** BTC

Input	Output
B′: **1.0** BTC	R: 0.5 BTC
	B‴: **0.5** BTC

Input	Output
B‴: **0.5** BTC	S: 0.7 BTC
B″: **0.2** BTC	

3. Mixing Sucks, Really!

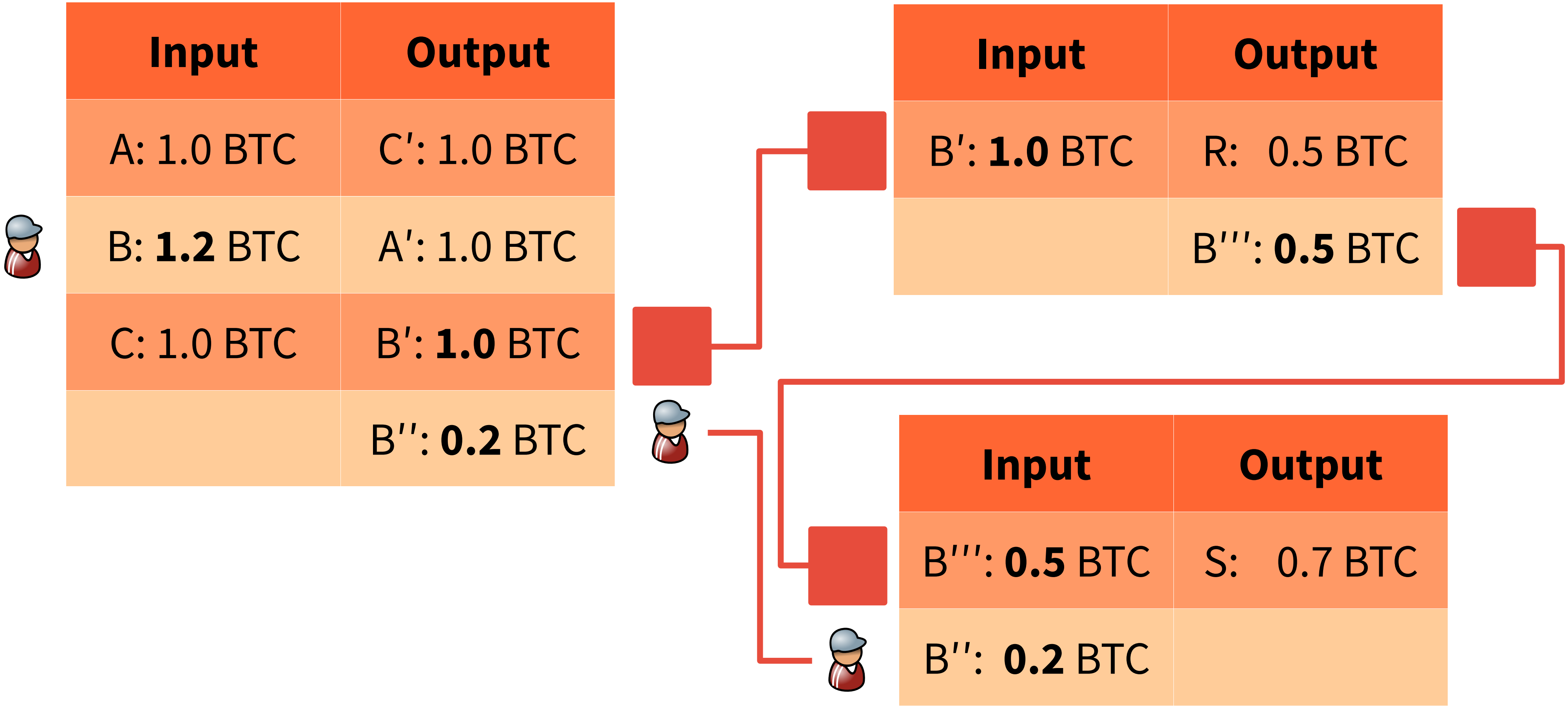


Input	Output
A: 1.0 BTC	C′: 1.0 BTC
B: **1.2** BTC	A′: 1.0 BTC
C: 1.0 BTC	B′: **1.0** BTC
	B″: **0.2** BTC

Input	Output
B′: **1.0** BTC	R: 0.5 BTC
	B‴: **0.5** BTC

Input	Output
B‴: **0.5** BTC	S: 0.7 BTC
B″: **0.2** BTC	

3. Mixing Sucks, Really!

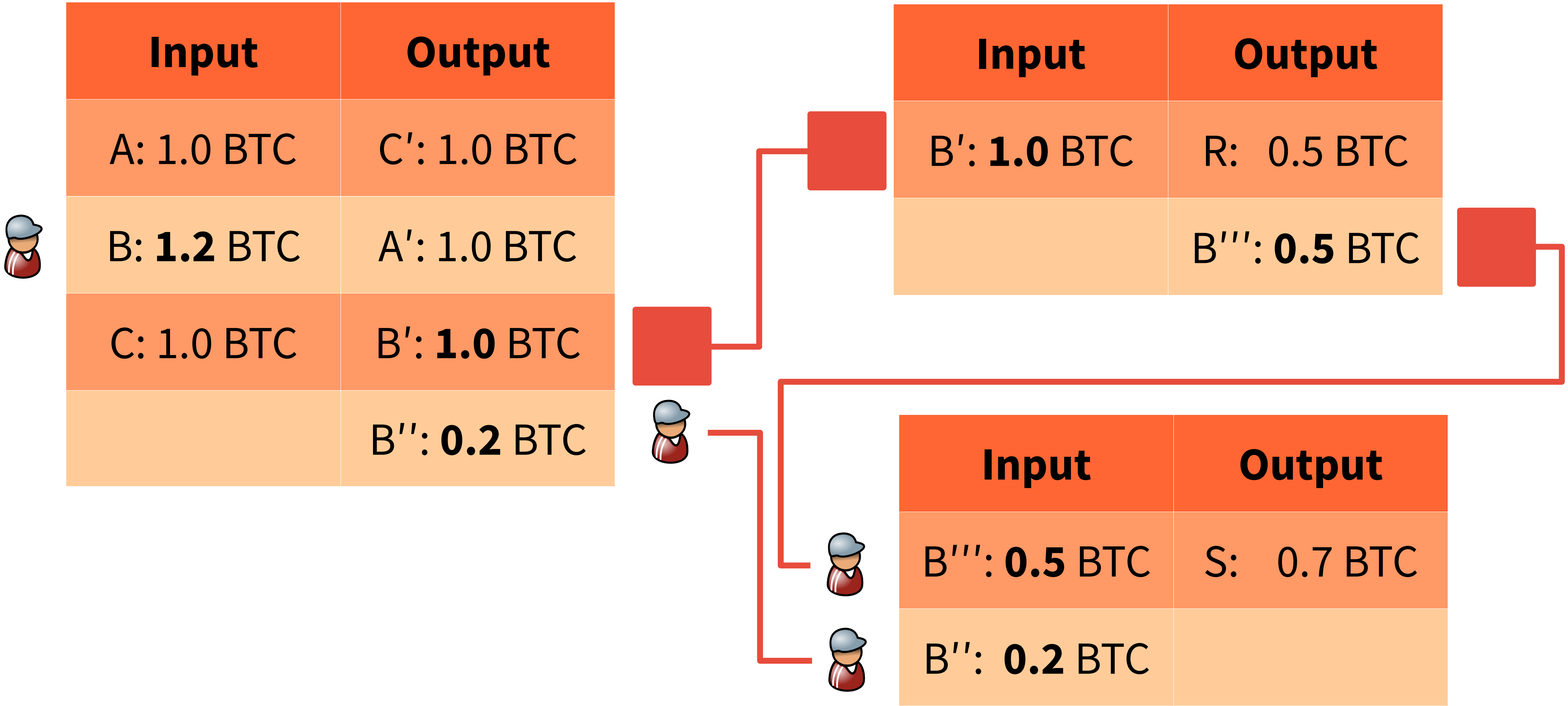


Input	Output
A: 1.0 BTC	C′: 1.0 BTC
B: **1.2** BTC	A′: 1.0 BTC
C: 1.0 BTC	B′: **1.0** BTC
	B″: **0.2** BTC

Input	Output
B′: **1.0** BTC	R: 0.5 BTC
	B‴: **0.5** BTC

Input	Output
B‴: **0.5** BTC	S: 0.7 BTC
B″: **0.2** BTC	

3. Mixing Sucks, Really!

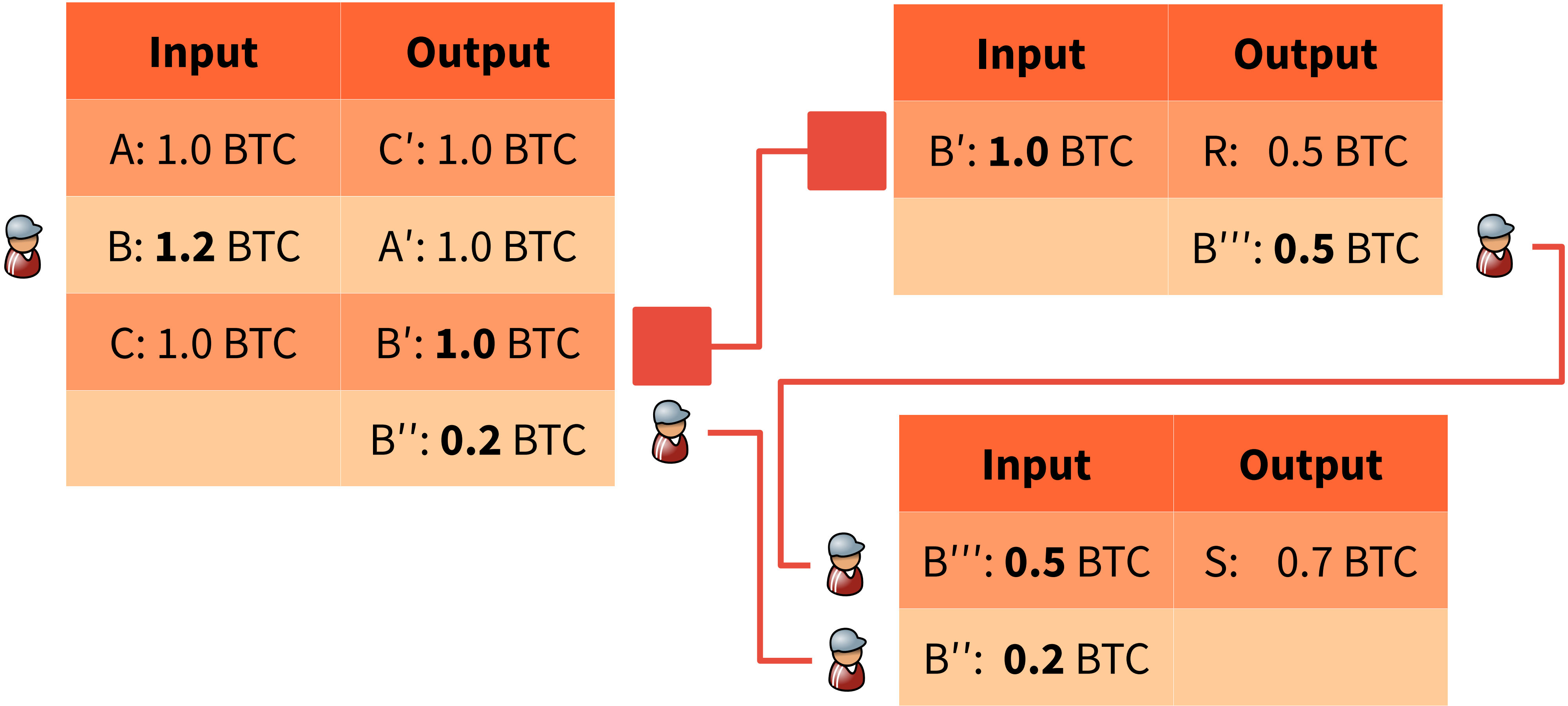


Input	Output
A: 1.0 BTC	C′: 1.0 BTC
B: **1.2** BTC	A′: 1.0 BTC
C: 1.0 BTC	B′: **1.0** BTC
	B′′: **0.2** BTC

Input	Output
B′: **1.0** BTC	R: 0.5 BTC
	B′′′: **0.5** BTC

Input	Output
B′′′: **0.5** BTC	S: 0.7 BTC
B′′: **0.2** BTC	

3. Mixing Sucks, Really!

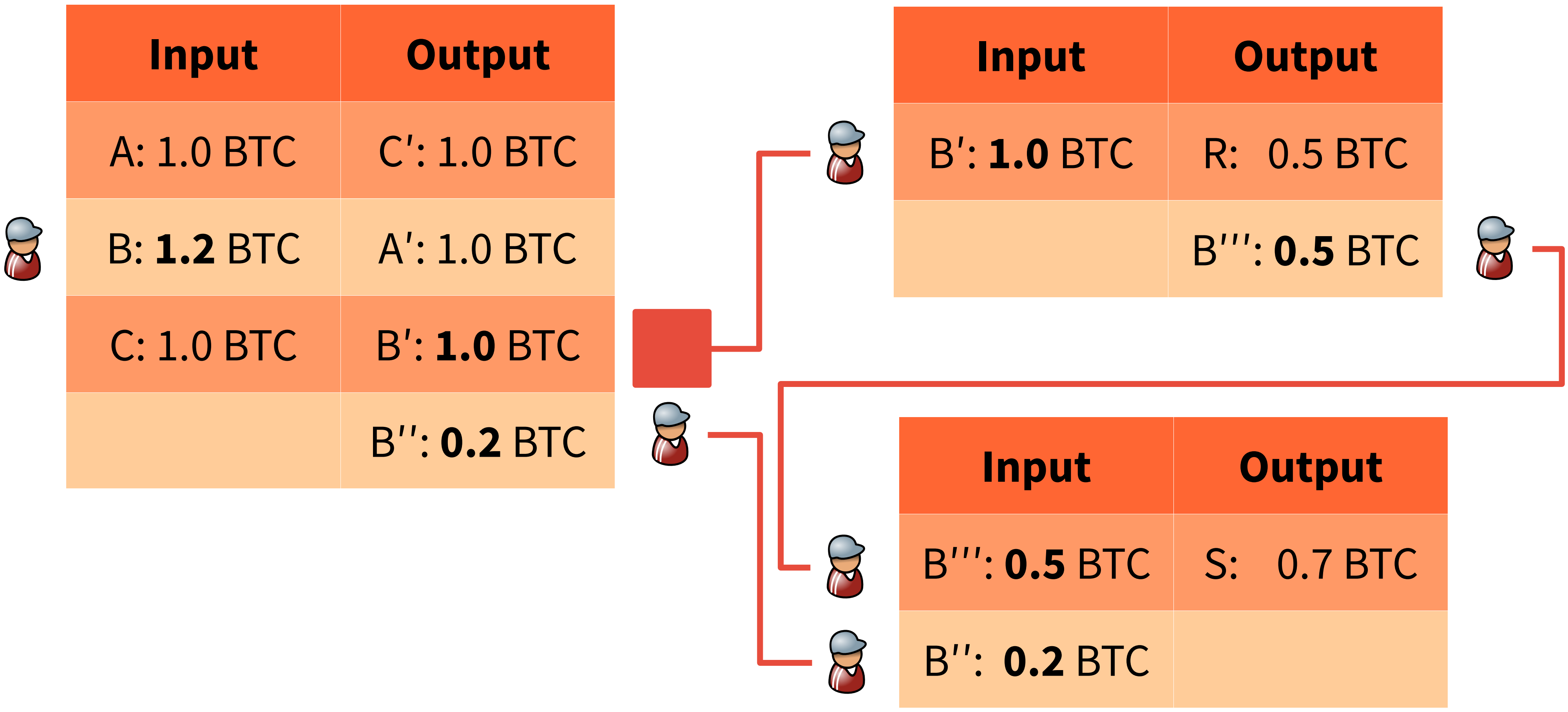


Input	Output
A: 1.0 BTC	C′: 1.0 BTC
B: **1.2** BTC	A′: 1.0 BTC
C: 1.0 BTC	B′: **1.0** BTC
	B″: **0.2** BTC

Input	Output
B′: **1.0** BTC	R: 0.5 BTC
	B‴: **0.5** BTC

Input	Output
B‴: **0.5** BTC	S: 0.7 BTC
B″: **0.2** BTC	

3. Mixing Sucks, Really!

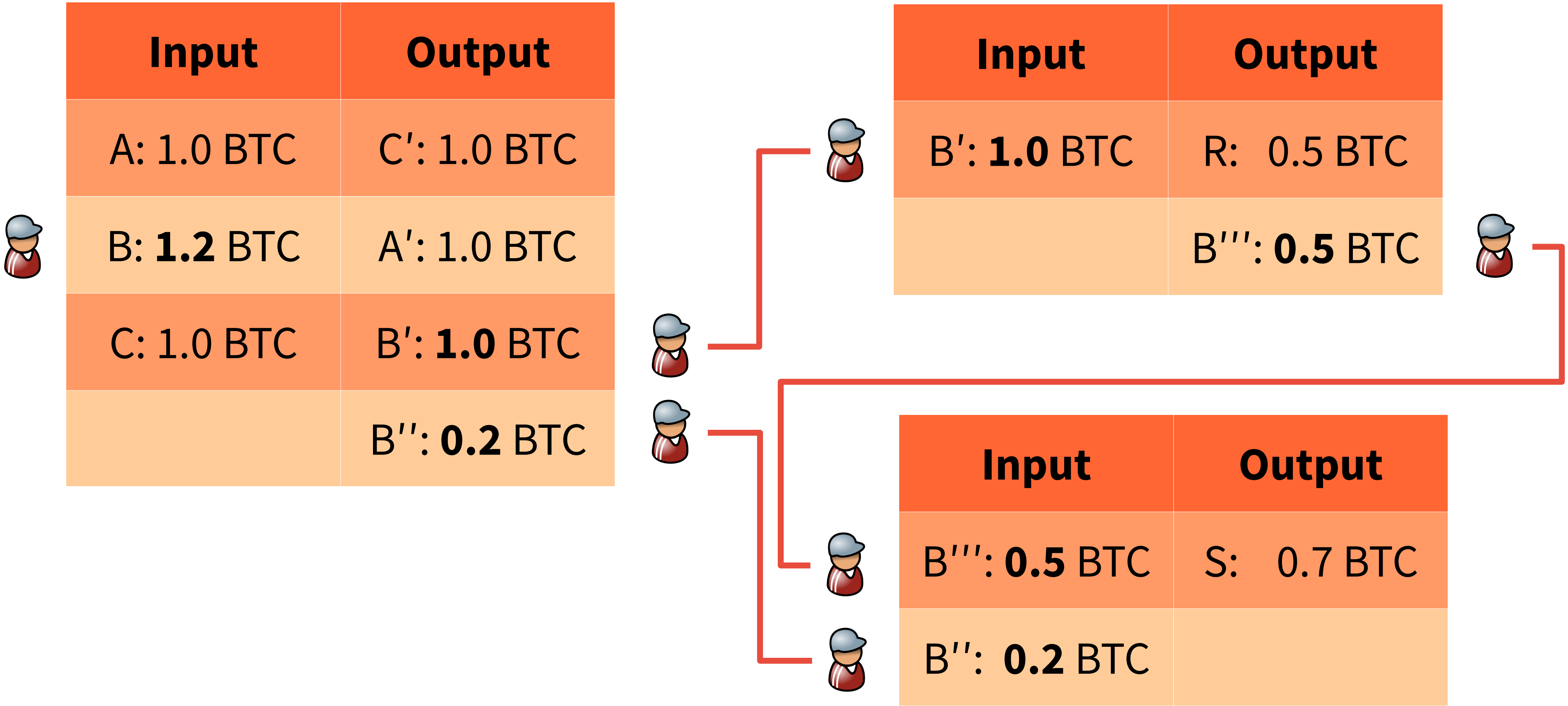


Input	Output
A: 1.0 BTC	C′: 1.0 BTC
B: **1.2** BTC	A′: 1.0 BTC
C: 1.0 BTC	B′: **1.0** BTC
	B″: **0.2** BTC

Input	Output
B′: **1.0** BTC	R: 0.5 BTC
	B‴: **0.5** BTC

Input	Output
B‴: **0.5** BTC	S: 0.7 BTC
B″: **0.2** BTC	

3. Mixing Sucks, Really!

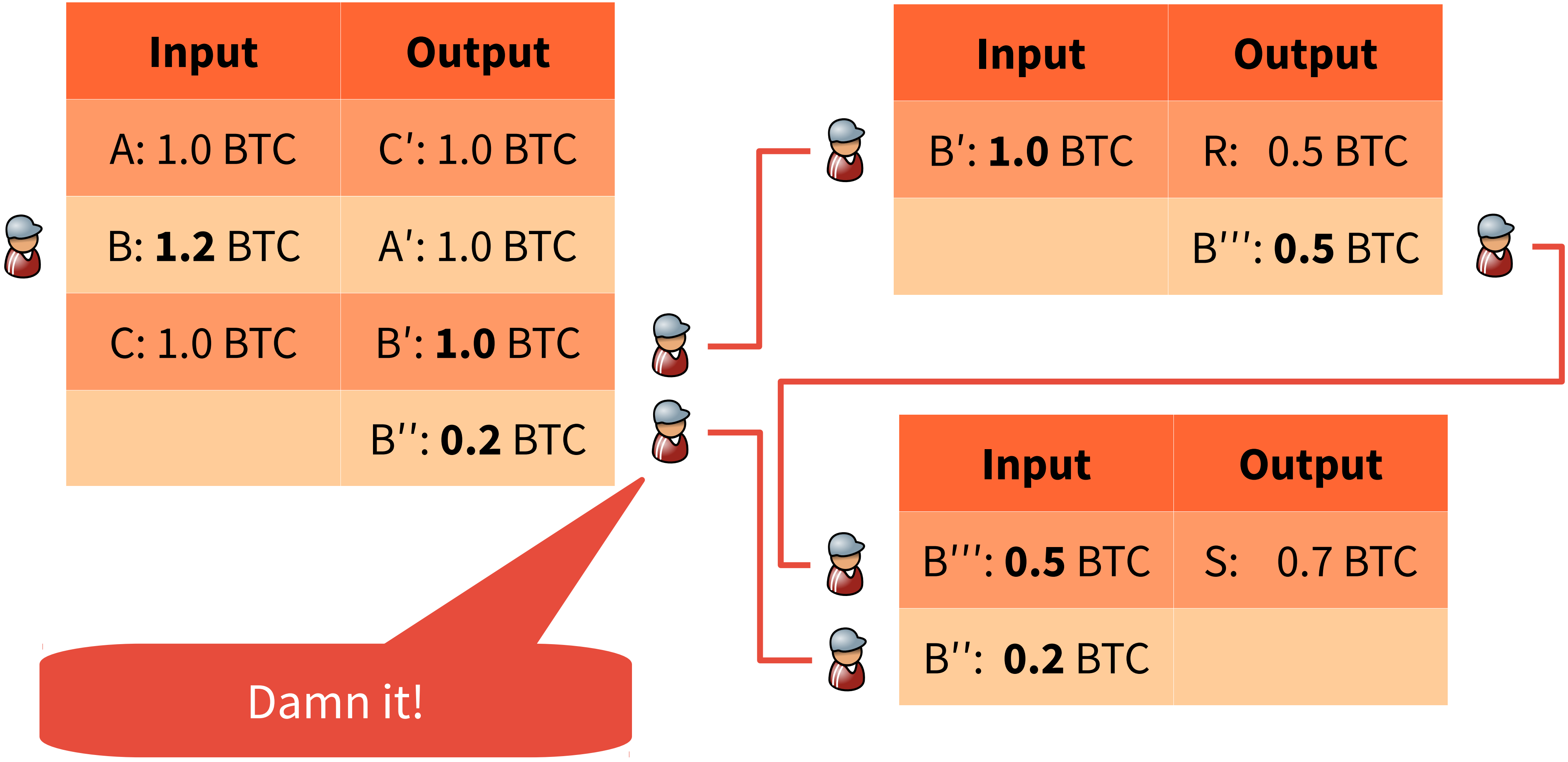


Input	Output
A: 1.0 BTC	C′: 1.0 BTC
B: **1.2** BTC	A′: 1.0 BTC
C: 1.0 BTC	B′: **1.0** BTC
	B′′: **0.2** BTC

Input	Output
B′: **1.0** BTC	R: 0.5 BTC
	B′′′: **0.5** BTC

Input	Output
B′′′: **0.5** BTC	S: 0.7 BTC
B′′: **0.2** BTC	

3. Mixing Sucks, Really!



Input	Output
A: 1.0 BTC	C′: 1.0 BTC
B: **1.2** BTC	A′: 1.0 BTC
C: 1.0 BTC	B′: **1.0** BTC
	B′′: **0.2** BTC

Input	Output
B′: **1.0** BTC	R: 0.5 BTC
	B′′′: **0.5** BTC

Input	Output
B′′′: **0.5** BTC	S: 0.7 BTC
B′′: **0.2** BTC	

Damn it!

WANTS TO MIX COINS
BUT GOT DEANONYMIZED
memegenerator.net

Many Problems

Root of all evil:
transacted values are public

Many Problems

Root of all evil:
transacted values are public

$$1.0 + 0.2 = 1.2$$

$$0.5 + 0.5 = 1.0$$

ValueShuffle

Let’s Add Confidential Transactions

Homomorphic Commitments

$$c = \mathrm{Com}(x, r)$$

Homomorphic Commitments

$$c = \mathrm{Com}(x, r)$$

- Hiding: given just *c*, you don't learn anything about *x*

Homomorphic Commitments

$$c = \mathrm{Com}(x, r)$$

- Hiding: given just *c*, you don't learn anything about *x*
- Binding: you cannot open *c* to anything but *x* (and create money)

Homomorphic Commitments

$$c = \mathrm{Com}(x, r)$$

- Hiding: given just *c*, you don't learn anything about *x*
- Binding: you cannot open *c* to anything but *x* (and create money)

$$\mathrm{Com}(x_1, r_1) + \mathrm{Com}(x_2, r_2) = \mathrm{Com}(x_1 + x_2, r_1 + r_2)$$

CoinJoin With Confidential Transactions

Input	Output
A: Com(5.4, $r_{\text{in},A}$)	C′: Com(0.1, $r_{\text{out},C'}$)
B: Com(1.2, $r_{\text{in},B}$)	B′: Com(0.7, $r_{\text{out},B'}$)
C: Com(0.3, $r_{\text{in},C}$)	RA: Com(0.4, $r_{\text{out},A}$)
	RC: Com(0.2, $r_{\text{out},C}$)
	A′: Com(5.0, $r_{\text{out},A'}$)
	RB: Com(0.5, $r_{\text{out},B}$)

CoinJoin With Confidential Transactions

+

Input	Output
A: $\mathrm{Com}(5.4, r_{\mathrm{in,A}})$	C′: $\mathrm{Com}(0.1, r_{\mathrm{out,C'}})$
B: $\mathrm{Com}(1.2, r_{\mathrm{in,B}})$	B′: $\mathrm{Com}(0.7, r_{\mathrm{out,B'}})$
C: $\mathrm{Com}(0.3, r_{\mathrm{in,C}})$	RA: $\mathrm{Com}(0.4, r_{\mathrm{out,A}})$
	RC: $\mathrm{Com}(0.2, r_{\mathrm{out,C}})$
	A′: $\mathrm{Com}(5.0, r_{\mathrm{out,A'}})$
	RB: $\mathrm{Com}(0.5, r_{\mathrm{out,B}})$

−

$\mathrm{Com}(0, r)$

CoinJoin With Confidential Transactions

Input	Output
A: Com(5.4, $r_{\text{in,A}}$)	C′: Com(0.1, $r_{\text{out,C'}}$)
B: Com(1.2, $r_{\text{in,B}}$)	B′: Com(0.7, $r_{\text{out,B'}}$)
C: Com(0.3, $r_{\text{in,C}}$)	RA: Com(0.4, $r_{\text{out,A}}$)
	RC: Com(0.2, $r_{\text{out,C}}$)
	A′: Com(5.0, $r_{\text{out,A'}}$)
	RB: Com(0.5, $r_{\text{out,B}}$)

$+$ $-$

Reveal excess value
to open the sum commitment

$$=$$

Com(0, r)

CoinJoin With Confidential Transactions

Input	Output
A: $\mathrm{Com}(5.4, r_{\mathrm{in},A})$	C′: $\mathrm{Com}(0.1, r_{\mathrm{out},C'})$
B: $\mathrm{Com}(1.2, r_{\mathrm{in},B})$	B′: $\mathrm{Com}(0.7, r_{\mathrm{out},B'})$
C: $\mathrm{Com}(0.3, r_{\mathrm{in},C})$	RA: $\mathrm{Com}(0.4, r_{\mathrm{out},A})$
	RC: $\mathrm{Com}(0.2, r_{\mathrm{out},C})$
	A′: $\mathrm{Com}(5.0, r_{\mathrm{out},A'})$
	RB: $\mathrm{Com}(0.5, r_{\mathrm{out},B})$

Reveal excess value
to open the sum commitment

$$\mathrm{Com}(0, r) = \mathrm{Com}(0, 0)$$

CoinJoin With Confidential Transactions

Input	Output
A: $\mathrm{Com}(5.4, r_{\mathrm{in},A})$	C′: $\mathrm{Com}(0.1, r_{\mathrm{out},C'})$
B: $\mathrm{Com}(1.2, r_{\mathrm{in},B})$	B′: $\mathrm{Com}(0.7, r_{\mathrm{out},B'})$
C: $\mathrm{Com}(0.3, r_{\mathrm{in},C})$	RA: $\mathrm{Com}(0.4, r_{\mathrm{out},A})$
	RC: $\mathrm{Com}(0.2, r_{\mathrm{out},C})$
	A′: $\mathrm{Com}(5.0, r_{\mathrm{out},A'})$
	RB: $\mathrm{Com}(0.5, r_{\mathrm{out},B})$

CoinJoin With Confidential Transactions

Input	Output
A: $\mathrm{Com}(5.4, r_{\text{in},A})$	C′: $\mathrm{Com}(0.1, r_{\text{out},C'})$
B: $\mathrm{Com}(1.2, \boxed{r_{\text{in},B}})$	B′: $\mathrm{Com}(0.7, \boxed{r_{\text{out},B'}})$
C: $\mathrm{Com}(0.3, r_{\text{in},C})$	RA: $\mathrm{Com}(0.4, r_{\text{out},A})$
	RC: $\mathrm{Com}(0.2, r_{\text{out},C})$
	A′: $\mathrm{Com}(5.0, r_{\text{out},A'})$
	RB: $\mathrm{Com}(0.5, \boxed{r_{\text{out},B}})$

CoinJoin With Confidential Transactions

Input	Output
A: $\mathrm{Com}(5.4, r_{\mathrm{in,A}})$	C′: $\mathrm{Com}(0.1, r_{\mathrm{out,C'}})$
B: $\mathrm{Com}(1.2, \boxed{r_{\mathrm{in,B}}})$	B′: $\mathrm{Com}(0.7, \boxed{r_{\mathrm{out,B'}}})$
C: $\mathrm{Com}(0.3, r_{\mathrm{in,C}})$	RA: $\mathrm{Com}(0.4, r_{\mathrm{out,A}})$
	RC: $\mathrm{Com}(0.2, r_{\mathrm{out,C}})$
	A′: $\mathrm{Com}(5.0, r_{\mathrm{out,A'}})$
	RB: $\mathrm{Com}(0.5, \boxed{r_{\mathrm{out,B}}})$

We need to compute the sum r
such that individual summands are not revealed.

CoinJoin With Confidential Transactions

Input	Output
A: Com(5.4, $r_{\text{in,A}}$)	C′: Com(0.1, $r_{\text{out,C}'}$)
B: Com(1.2, $r_{\text{in,B}}$)	B′: Com(0.7, $r_{\text{out,B}'}$)
C: Com(0.3, $r_{\text{in,C}}$)	RA: Com(0.4, $r_{\text{out,A}}$)
	RC: Com(0.2, $r_{\text{out,C}}$)
	A′: Com(5.0, $r_{\text{out,A}'}$)
	RB: Com(0.5, $r_{\text{out,B}}$)
	F: 0.0, $-\boldsymbol{r}$

CoinJoin With Confidential Transactions

+

Input	Output
A: $\mathrm{Com}(5.4, r_{\mathrm{in},A})$	C′: $\mathrm{Com}(0.1, r_{\mathrm{out},C'})$
B: $\mathrm{Com}(1.2, r_{\mathrm{in},B})$	B′: $\mathrm{Com}(0.7, r_{\mathrm{out},B'})$
C: $\mathrm{Com}(0.3, r_{\mathrm{in},C})$	RA: $\mathrm{Com}(0.4, r_{\mathrm{out},A})$
	RC: $\mathrm{Com}(0.2, r_{\mathrm{out},C})$
	A′: $\mathrm{Com}(5.0, r_{\mathrm{out},A'})$
	RB: $\mathrm{Com}(0.5, r_{\mathrm{out},B})$
	F: 0.0, $-\boldsymbol{r}$

−

$\mathrm{Com}(0, 0)$

ValueShuffle

ValueShuffle = DiceMix + Secure Sum Protocol

Sanity Check: Are Messages Discardable?

Input	Output
A: Com(5.4, $r_{in,A}$)	C′: Com(0.1, $r_{out,C'}$)
B: Com(1.2, $r_{in,B}$)	B′: Com(0.7, $r_{out,B'}$)
C: Com(0.3, $r_{in,C}$)	RA: Com(0.4, $r_{out,A}$)
	RC: Com(0.2, $r_{out,C}$)
	A′: Com(5.0, $r_{out,A'}$)
	RB: Com(0.5, $r_{out,B}$)

Sanity Check: Are Messages Discardable?

Input	Output
A: $\mathrm{Com}(5.4, r_{\mathrm{in},A})$	C′: $\mathrm{Com}(0.1, r_{\mathrm{out},C'})$
B: $\mathrm{Com}(1.2, r_{\mathrm{in},B})$	B′: $\mathrm{Com}(0.7, r_{\mathrm{out},B'})$
C: $\mathrm{Com}(0.3, r_{\mathrm{in},C})$	RA: $\mathrm{Com}(0.4, r_{\mathrm{out},A})$
	RC: $\mathrm{Com}(0.2, r_{\mathrm{out},C})$
	A′: $\mathrm{Com}(5.0, r_{\mathrm{out},A'})$
	RB: $\mathrm{Com}(0.5, r_{\mathrm{out},B})$

Bob’s messages in mixing protocol:
(B′, $\mathrm{Com}(0.7, r_{\mathrm{out},B'})$, $\mathrm{aux}_{B'}$) and (RB, $\mathrm{Com}(0.5, r_{\mathrm{out},B})$, aux_{RB})

Sanity Check: Are Messages Discardable?

Input	Output
A: $\mathrm{Com}(5.4, r_{\mathrm{in},A})$	C′: $\mathrm{Com}(0.1, r_{\mathrm{out},C'})$
B: $\mathrm{Com}(1.2, r_{\mathrm{in},B})$	B′: $\mathrm{Com}(0.7, r_{\mathrm{out},B'})$
C: $\mathrm{Com}(0.3, r_{\mathrm{in},C})$	RA: $\mathrm{Com}(0.4, r_{\mathrm{out},A})$
	RC: $\mathrm{Com}(0.2, r_{\mathrm{out},C})$
	A′: $\mathrm{Com}(5.0, r_{\mathrm{out},A'})$
	RB: $\mathrm{Com}(0.5, r_{\mathrm{out},B})$

Discardable change address

Bob’s messages in mixing protocol:

(B′, $\mathrm{Com}(0.7, r_{\mathrm{out},B'})$, $\mathrm{aux}_{B'}$) and (RB, $\mathrm{Com}(0.5, r_{\mathrm{out},B})$, aux_{RB})

Sanity Check: Are Messages Discardable?

Input	Output
A: Com(5.4, $r_{in,A}$)	C′: Com(0.1, $r_{out,C'}$)
B: Com(1.2, $r_{in,B}$)	B′: Com(0.7, $r_{out,B'}$)
C: Com(0.3, $r_{in,C}$)	RA: Com(0.4, $r_{out,A}$)
	RC: Com(0.2, $r_{out,C}$)
	: Com(5.0, $r_{out,A'}$)
	RB: Com(0.5, $r_{out,B}$)

Discardable commitments

Bob’s messages in mixing protocol:
(B′, Com(0.7, $r_{out,B'}$), $\mathrm{aux}_{B'}$) and (RB, Com(0.5, $r_{out,B}$), aux_{RB})

Sanity Check: Are Messages Discardable?

Input	Output
A: Com(5.4, $r_{in,A}$)	C′: Com(0.1, $r_{out,C'}$)
B: Com(1.2, $r_{in,B}$)	B′: Com(0.7, $r_{out,B'}$)
C: Com(0.3, $r_{in,C}$)	RA: Com(0.4, $r_{out,A}$)
	RC: Com(0.2, $r_{out,C}$)
	RB: Com(0. $_{B}$)

Discardable aux info
(range proofs)

Bob's messages in mixing protocol:
(B′, Com(0.7, $r_{out,B'}$), $aux_{B'}$) and (RB, Com(0.5, $r_{out,B}$), aux_{RB})

Sanity Check: Are Messages Discardable?

Input	Output
A: $\mathrm{Com}(5.4, r_{in,A})$	C′: $\mathrm{Com}(0.1, r_{out,C'})$
B: $\mathrm{Com}(1.2, r_{in,B})$	B′: $\mathrm{Com}(0.7, r_{out,B'})$
C: $\mathrm{Com}(0.3, r_{in,C})$	RA: $\mathrm{Com}(0.4, r_{out,A})$
	RC: $\mathrm{Com}(0.2, r_{out,C})$
	RB: (, $r_{out,B}$)

Discardable recipient address
(BIP 32, stealth addresses, …)

Bob’s messages in mixing protocol:
(B′, $\mathrm{Com}(0.7, r_{out,B'})$, $\mathrm{aux}_{B'}$) and (RB, $\mathrm{Com}(0.5, r_{out,B})$, aux_{RB})

CoinJoin as It Should Be

CoinJoin as It Should Be

- No problems with change addresses ✓

CoinJoin as It Should Be

- No problems with change addresses ✓
- No need for two transactions to spend ✓

CoinJoin as It Should Be

- No problems with change addresses ✓
- No need for two transactions to spend ✓
- No foot-cannon when spending change ✓

CoinJoin as It Should Be

- No problems with change addresses ✓
- No need for two transactions to spend ✓
- No foot-cannon when spending change ✓
- No need to have the same amounts ✓

CoinJoin as It Should Be

- No problems with change addresses ✓
- No need for two transactions to spend ✓
- No foot-cannon when spending change ✓
- No need to have the same amounts ✓

Great synergy:
value privacy and unlinkability

Privacy Pays Off!

Privacy Pays Off!

- CoinJoin transaction smaller than set of individual transactions

Privacy Pays Off!

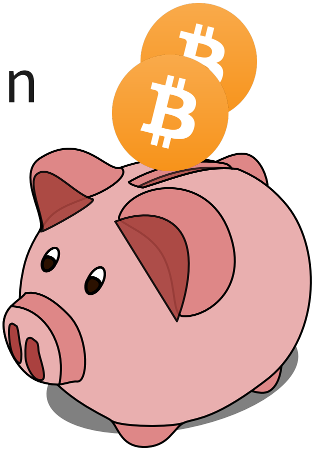
- CoinJoin transaction smaller than set of individual transactions
- **Really takes off with signature aggregation (e.g. Bellare-Neven)**

Privacy Pays Off!

- CoinJoin transaction smaller than set of individual transactions
- **Really takes off with signature aggregation (e.g. Bellare-Neven)**
- We save
  - Precious space in the blockchain
  - Verification time

Privacy Pays Off!

- CoinJoin transaction smaller than set of individual transactions
- **Really takes off with signature aggregation (e.g. Bellare-Neven)**
- We save
  - Precious space in the blockchain
  - Verification time
- User saves fees!



WANTS TO MIX COINS
AND SAVES FEES!
memegenerator.net

Variants of DiceMix

DiceMix

- $4 + 2f$ communication rounds
- Some heavy computation if messages are large (Polynomial factorization in finite fields)
- Variant in the paper

Variants of DiceMix

DiceMix

- 4 + 2*f* communication rounds
- Some heavy computation if messages are large (Polynomial factorization in finite fields)
- Variant in the paper

DiceMix Light

- 5 + 3*f* communication rounds
- No heavy computation
- Simpler protocol
- https://github.com/ElementsProject/dicemix

Practical Issues

Practical Issues

- Banning disruptive users

Practical Issues

- Banning disruptive users
  - Naive approach: Server keeps a ban list of disruptive users (= UTXOs)

Practical Issues

- Banning disruptive users
  - Naive approach: Server keeps a ban list of disruptive users (= UTXOs)
  - Current idea: Just rate limiting (*n* tries per UTXO) like JoinMarket

Practical Issues

- Banning disruptive users
  - Naive approach: Server keeps a ban list of disruptive users (= UTXOs)
  - Current idea: Just rate limiting (*n* tries per UTXO) like JoinMarket
  - Public key *pk* gives you *n* tokens

Practical Issues

- Banning disruptive users
  - Naive approach: Server keeps a ban list of disruptive users (= UTXOs)
  - Current idea: Just rate limiting (*n* tries per UTXO) like JoinMarket
  - Public key *pk* gives you *n* tokens
  - Users need to add a token to the ban list when starting a protocol

Practical Issues

- Banning disruptive users
  - Naive approach: Server keeps a ban list of disruptive users (= UTXOs)
  - Current idea: Just rate limiting (*n* tries per UTXO) like JoinMarket
  - Public key *pk* gives you *n* tokens
  - Users need to add a token to the ban list when starting a protocol
- Double-spending

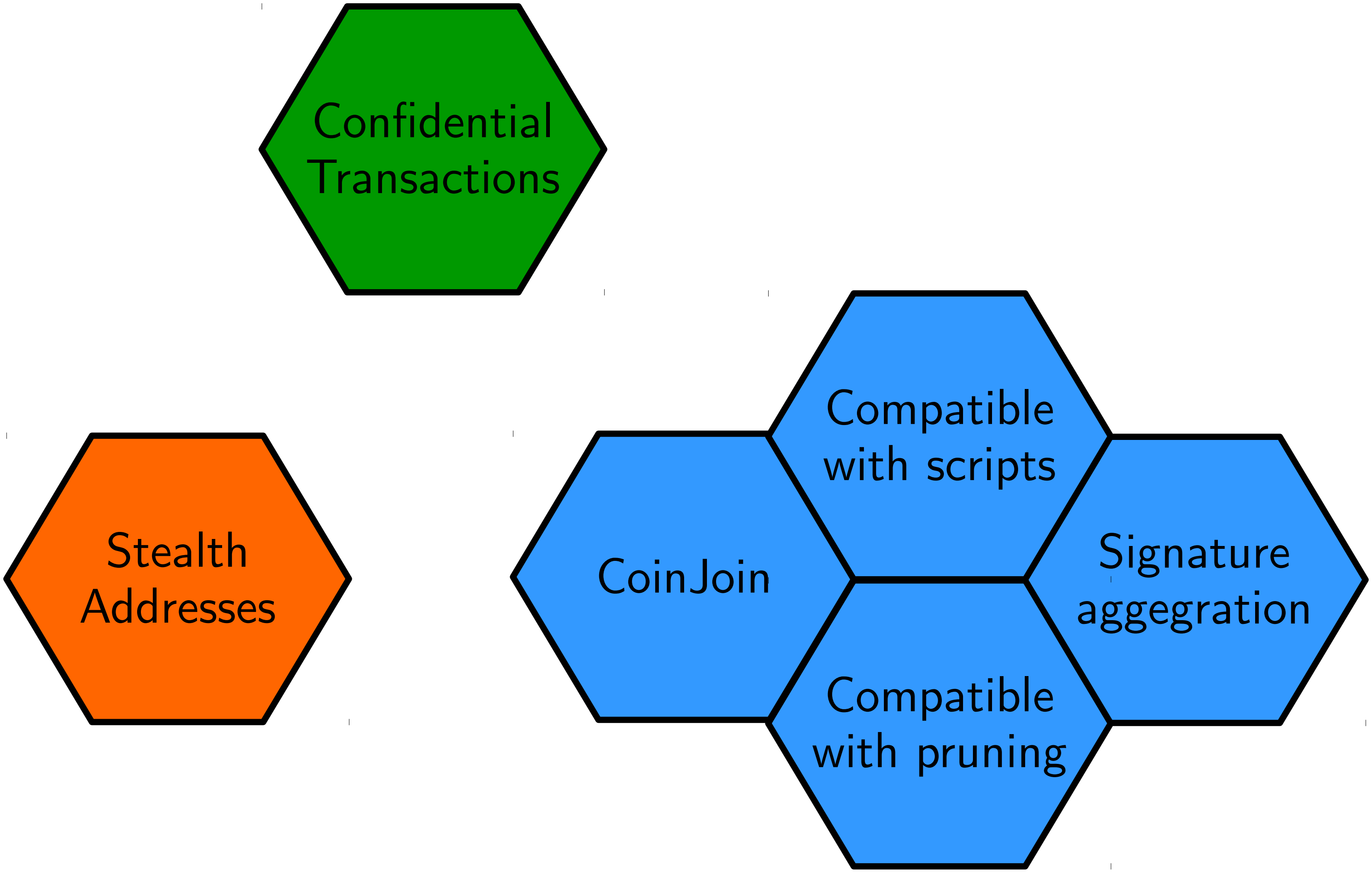
Practical Issues

- Banning disruptive users
  - Naive approach: Server keeps a ban list of disruptive users (= UTXOs)
  - Current idea: Just rate limiting (*n* tries per UTXO) like JoinMarket
  - Public key *pk* gives you *n* tokens
  - Users need to add a token to the ban list when starting a protocol
- Double-spending
- Availability of bulletin board

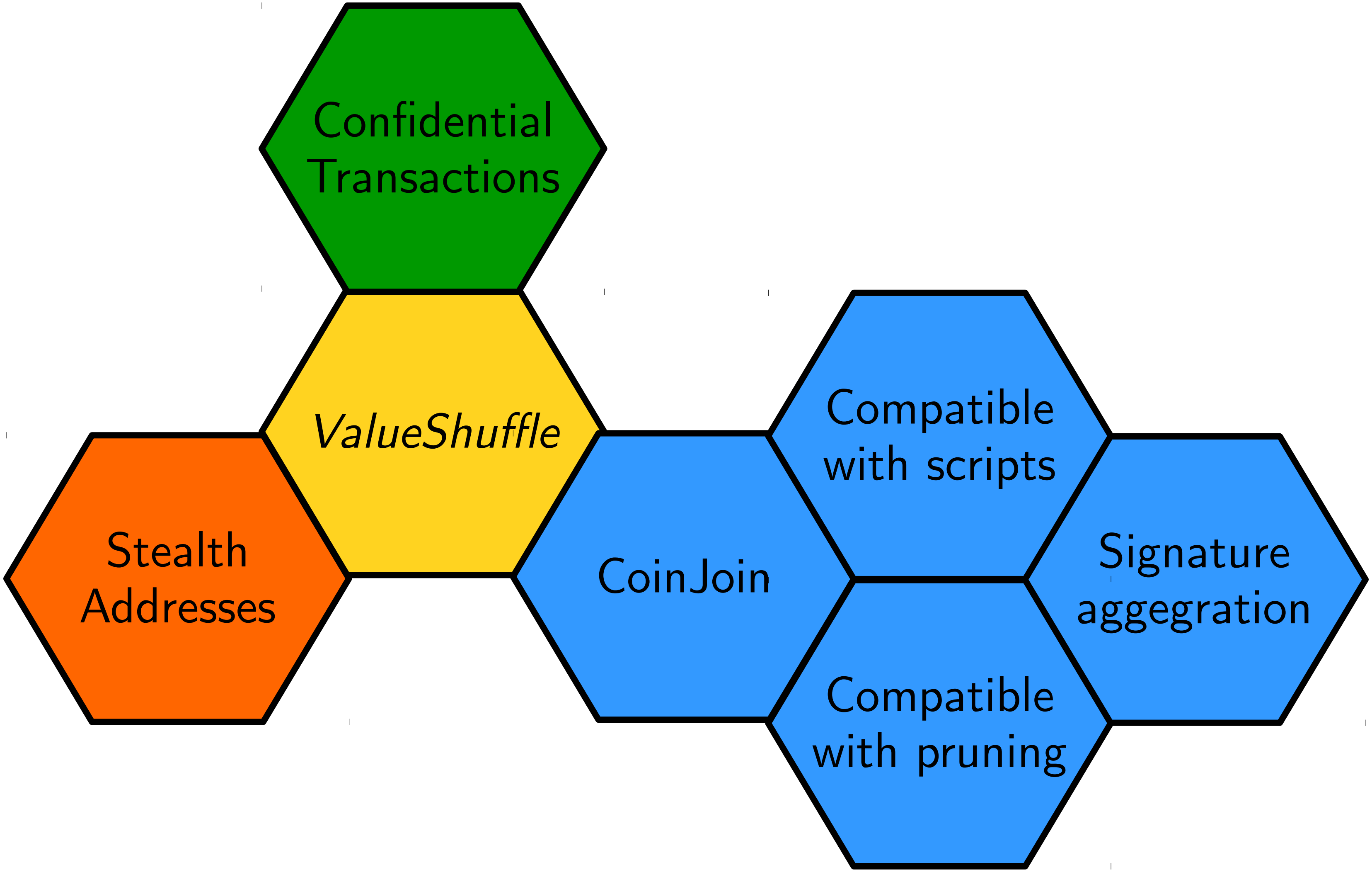
Practical Issues

- Banning disruptive users
  - Naive approach: Server keeps a ban list of disruptive users (= UTXOs)
  - Current idea: Just rate limiting (*n* tries per UTXO) like JoinMarket
  - Public key *pk* gives you *n* tokens
  - Users need to add a token to the ban list when starting a protocol
- Double-spending
- Availability of bulletin board
- Other issues?

ValueShuffle in the Bitcoin Privacy Landscape

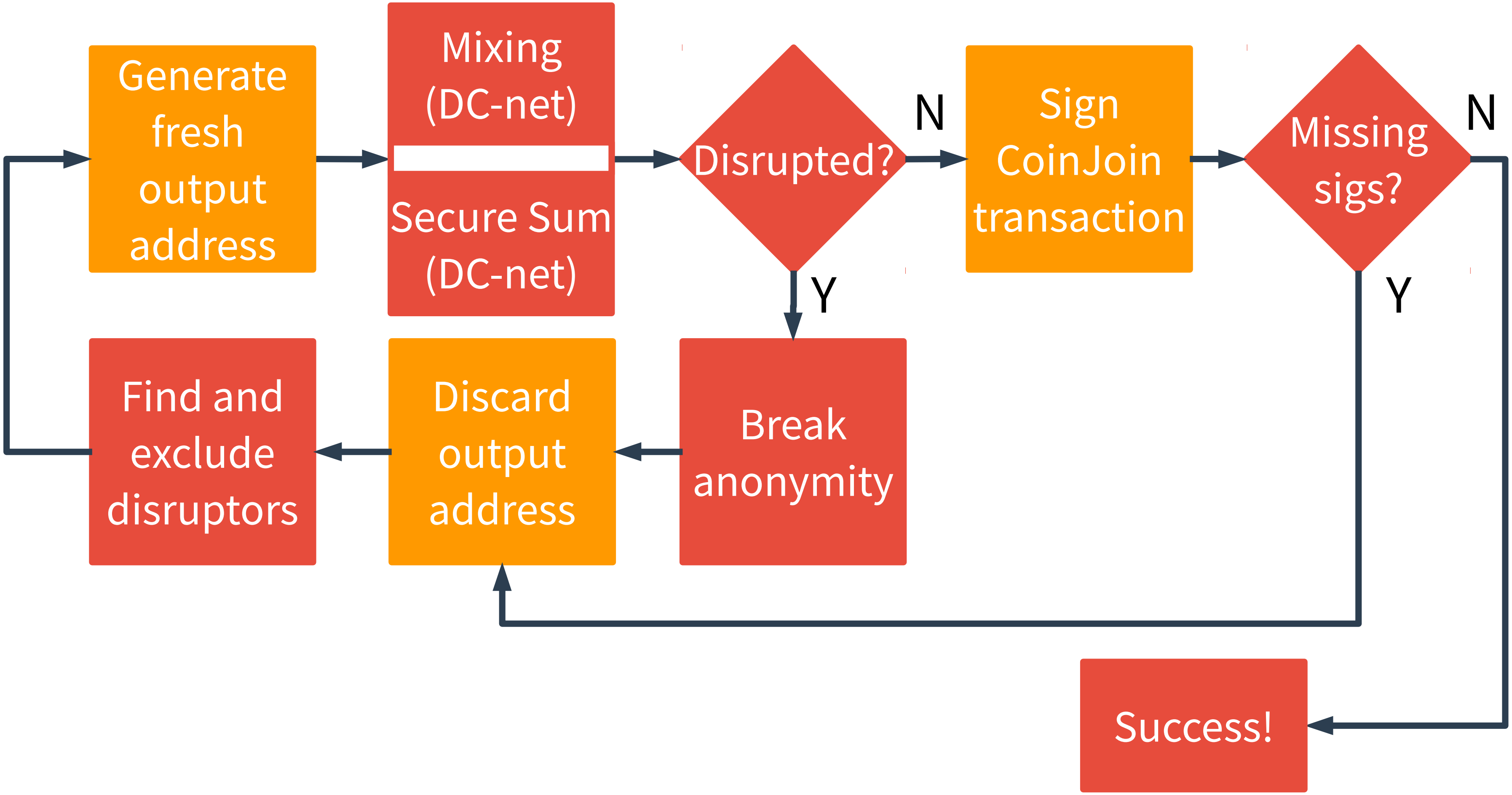


ValueShuffle in the Bitcoin Privacy Landscape

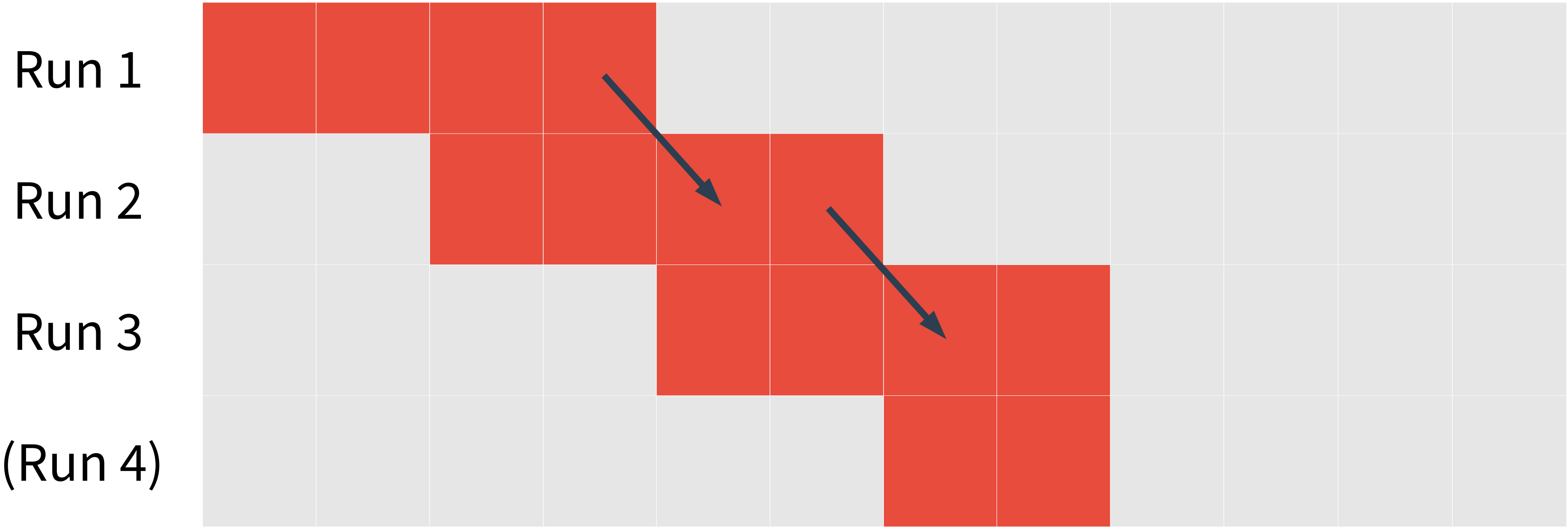


Backup Slides

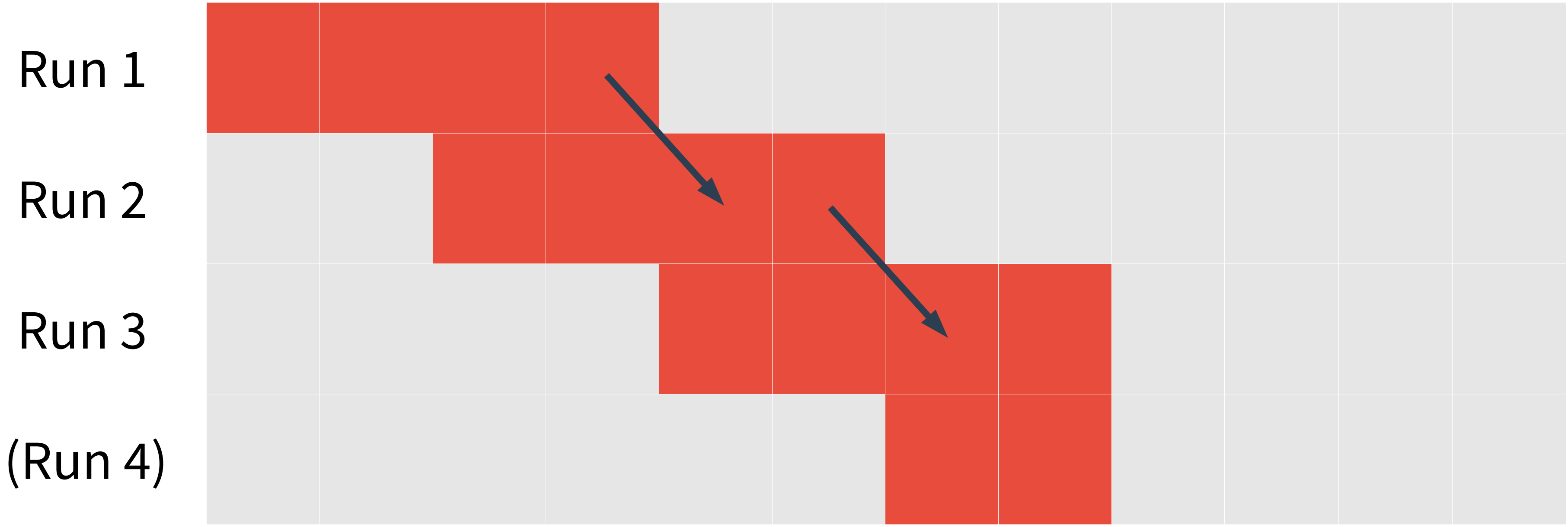
Flowchart of ValueShuffle



Performance



Performance



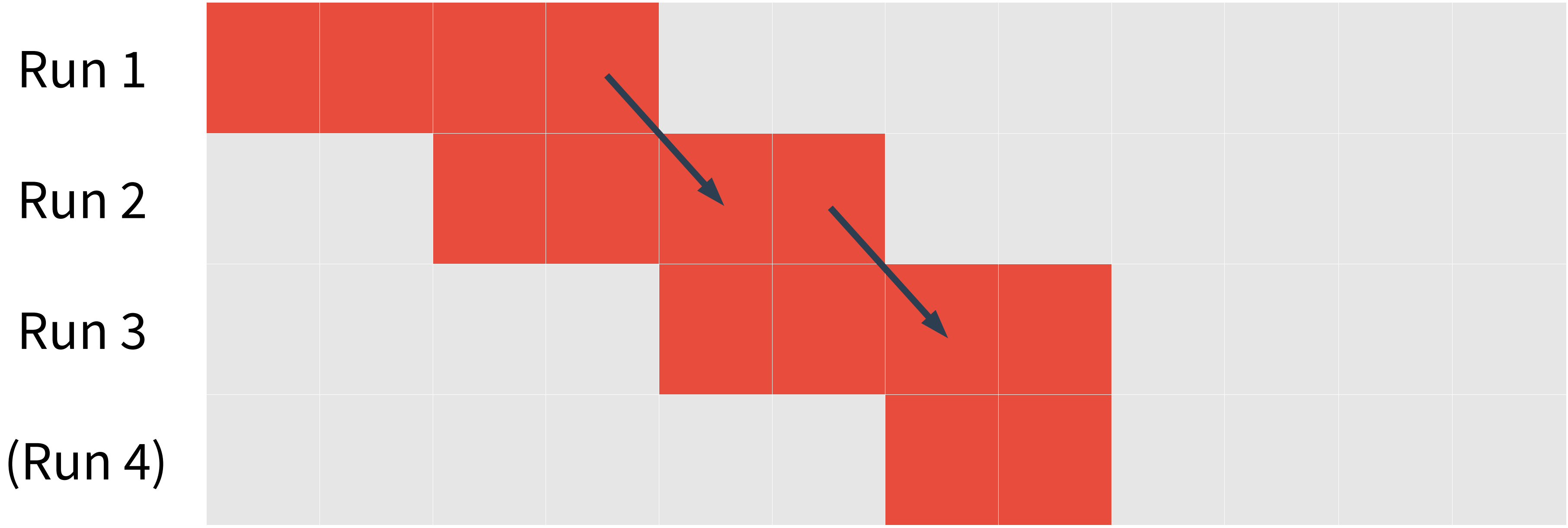
Run 1

Run 2

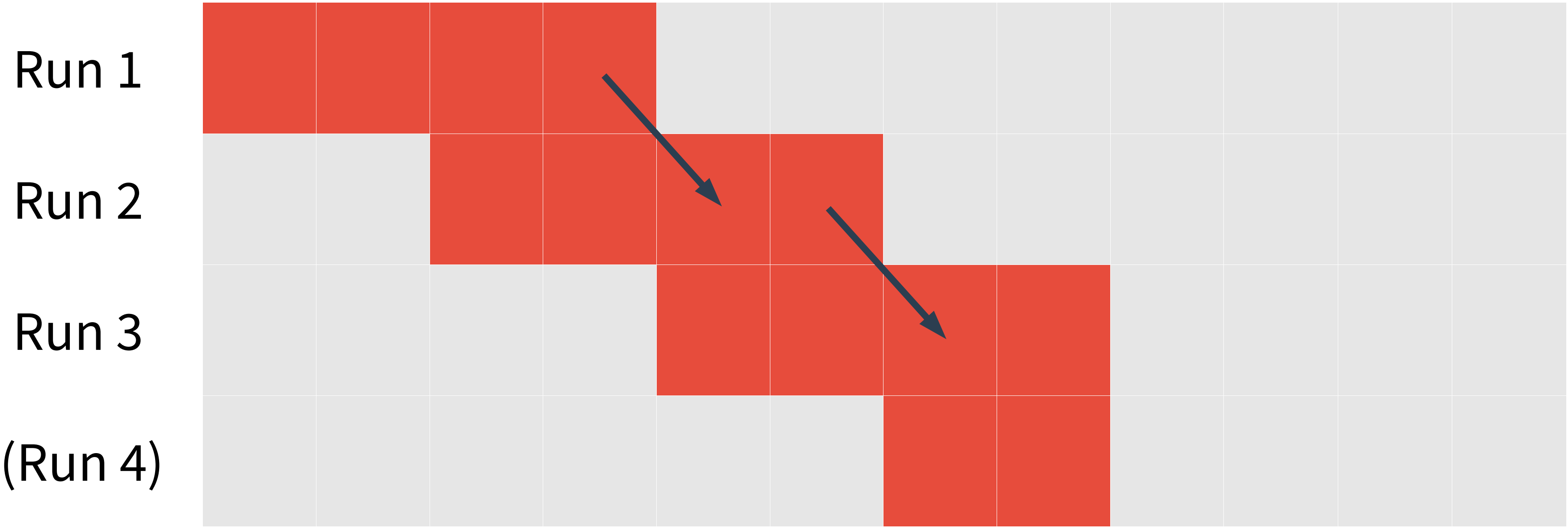
Run 3

(Run 4)

Performance



Performance



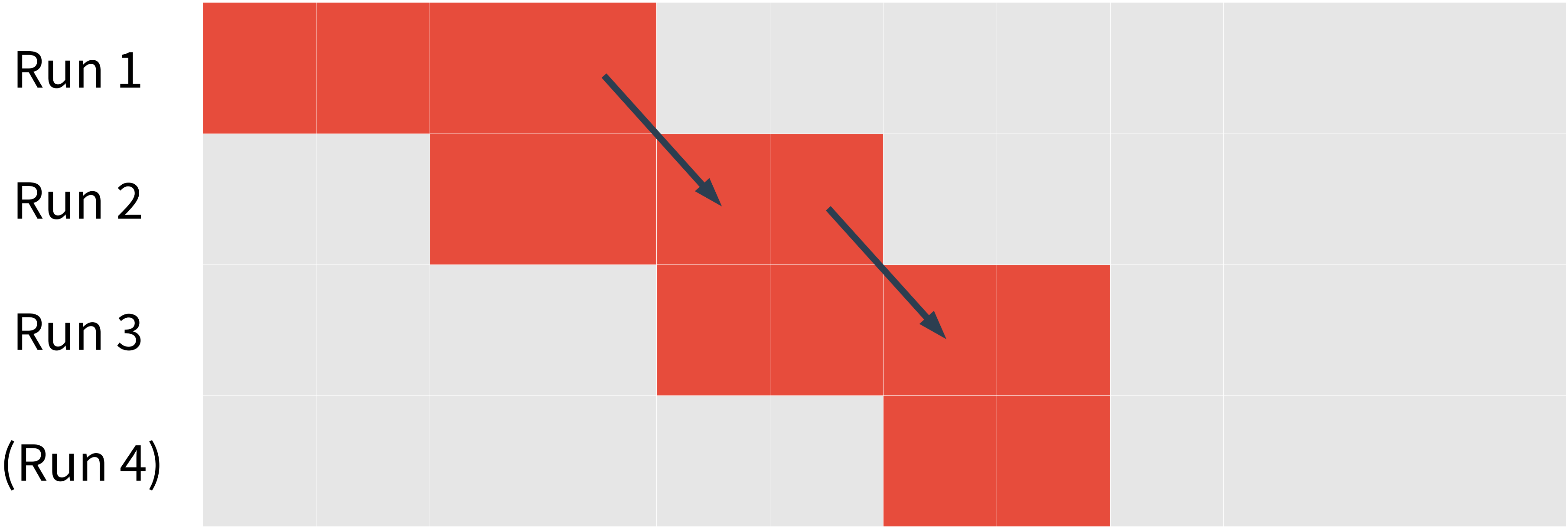
Run 1

Run 2

Run 3

(Run 4)

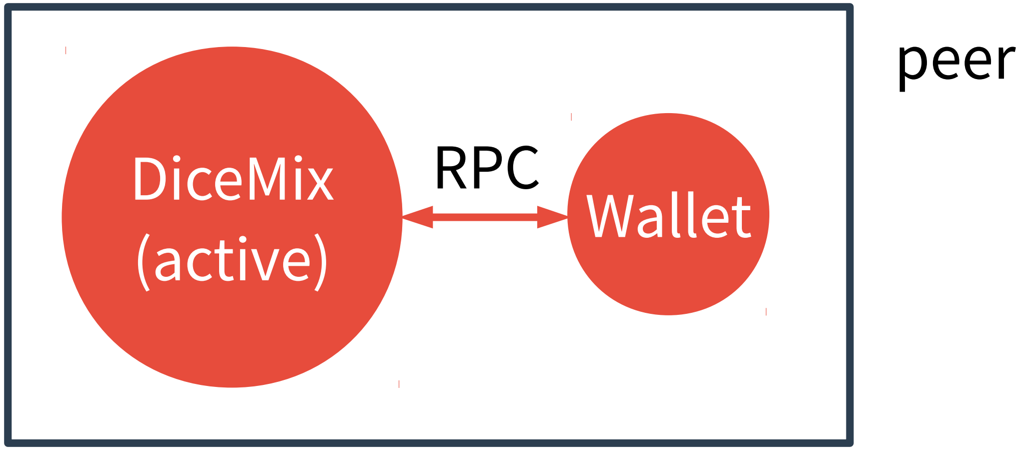
Performance



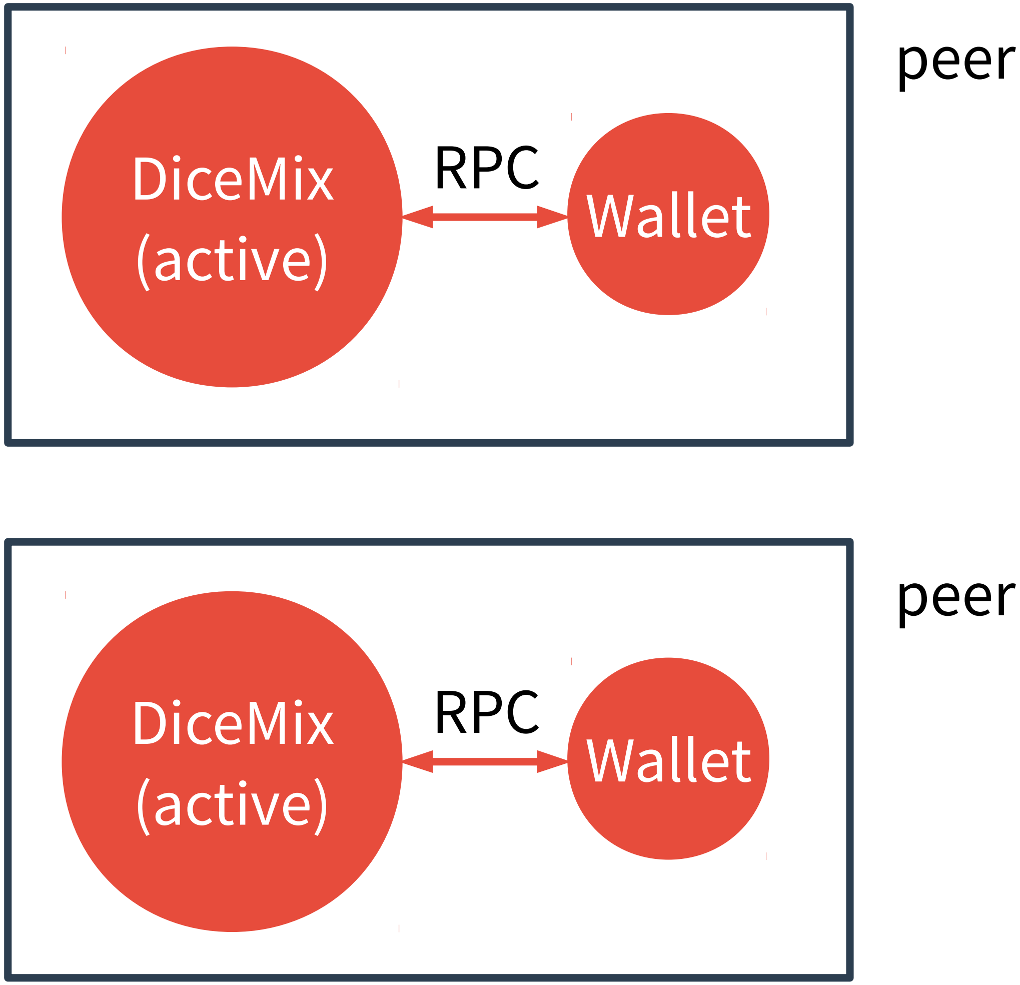
$4 + 2f$ rounds

(f disrupting peers)

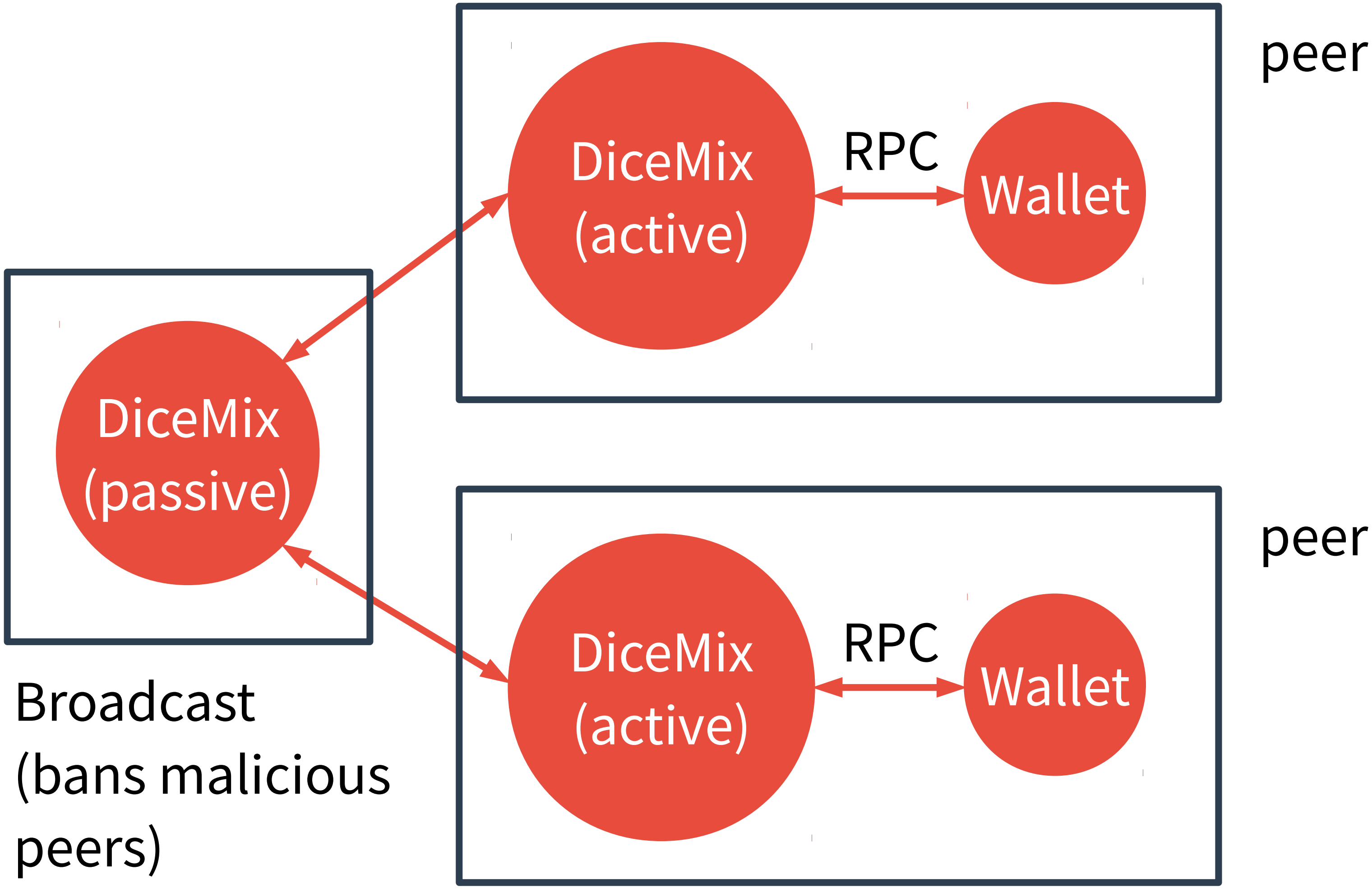
Architecture



Architecture



Architecture

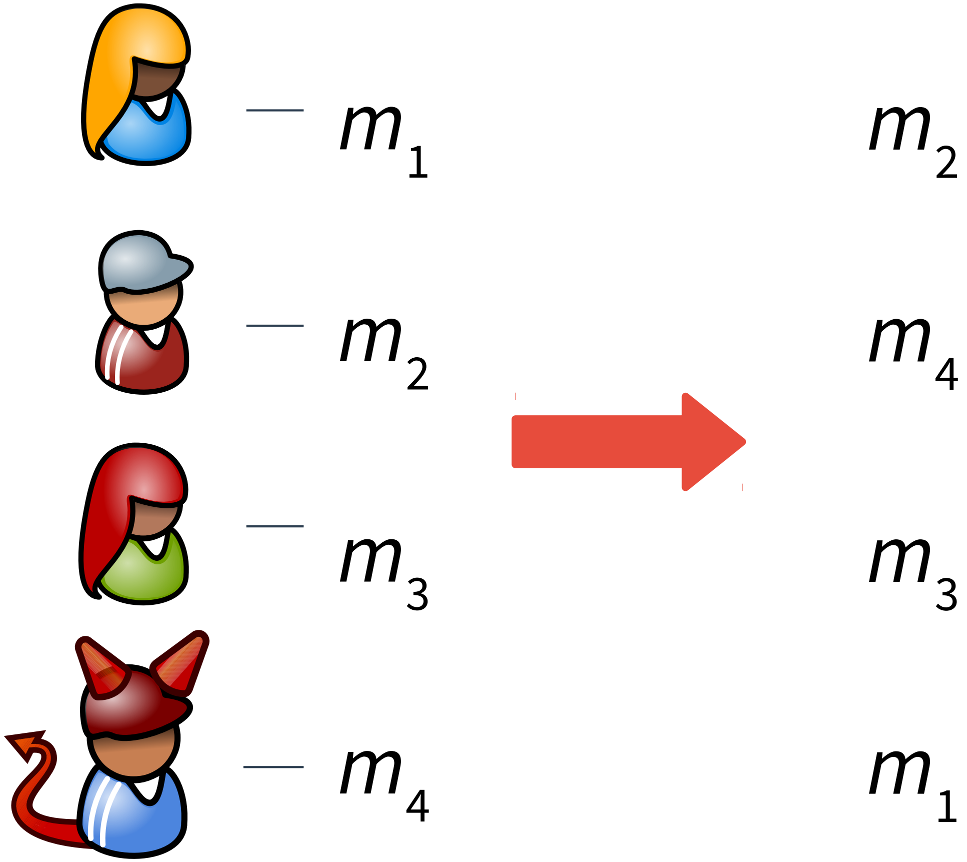


Broadcast (bans malicious peers)

Comparison with Related Work

	Anonymity set	Mixing overhead	Non-interactive	Pruning
ValueShuffle	Moderate (~ 50)	off-chain	no	yes
Monero / Ring-CT	Small (~ 10)	on-chain	yes	no
TumbleBit	Large (~ 800)	4 tx per mixing (classic mode)	yes	yes
Zcash	Full	?	yes	no

Idea of Attack on Anonymity



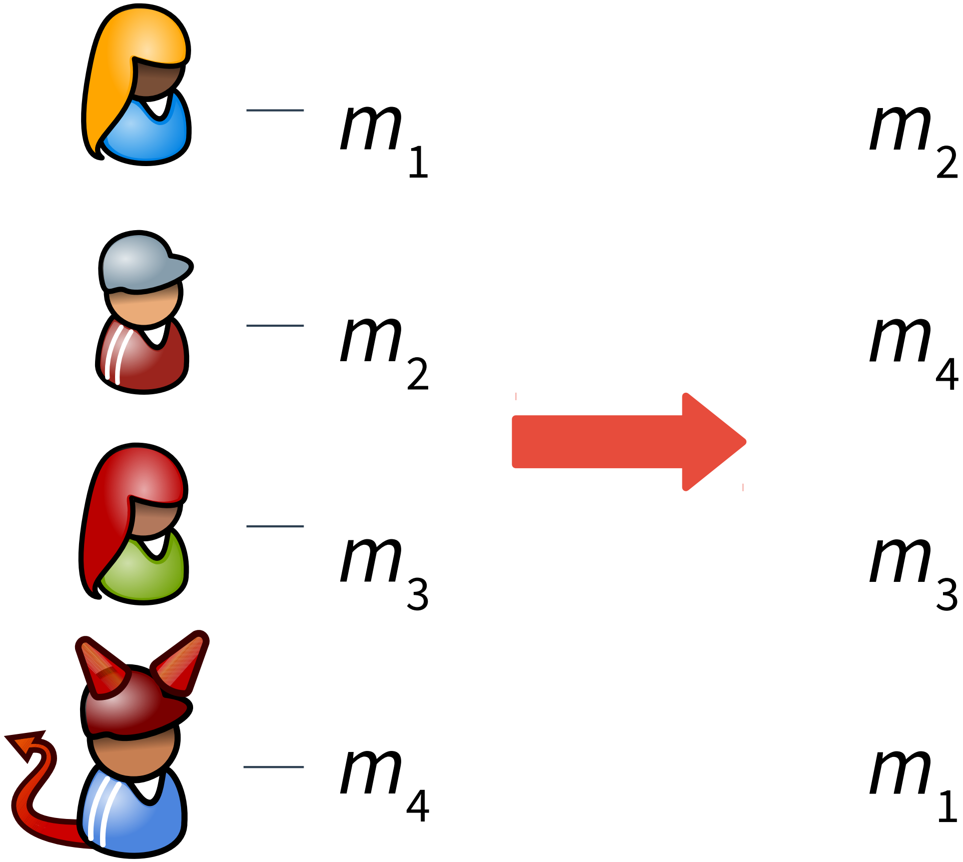
m_1 m_2

m_2 m_4

m_3 m_3

m_4 m_1

Idea of Attack on Anonymity



— m_1

— m_2

— m_3

— m_4

m_2

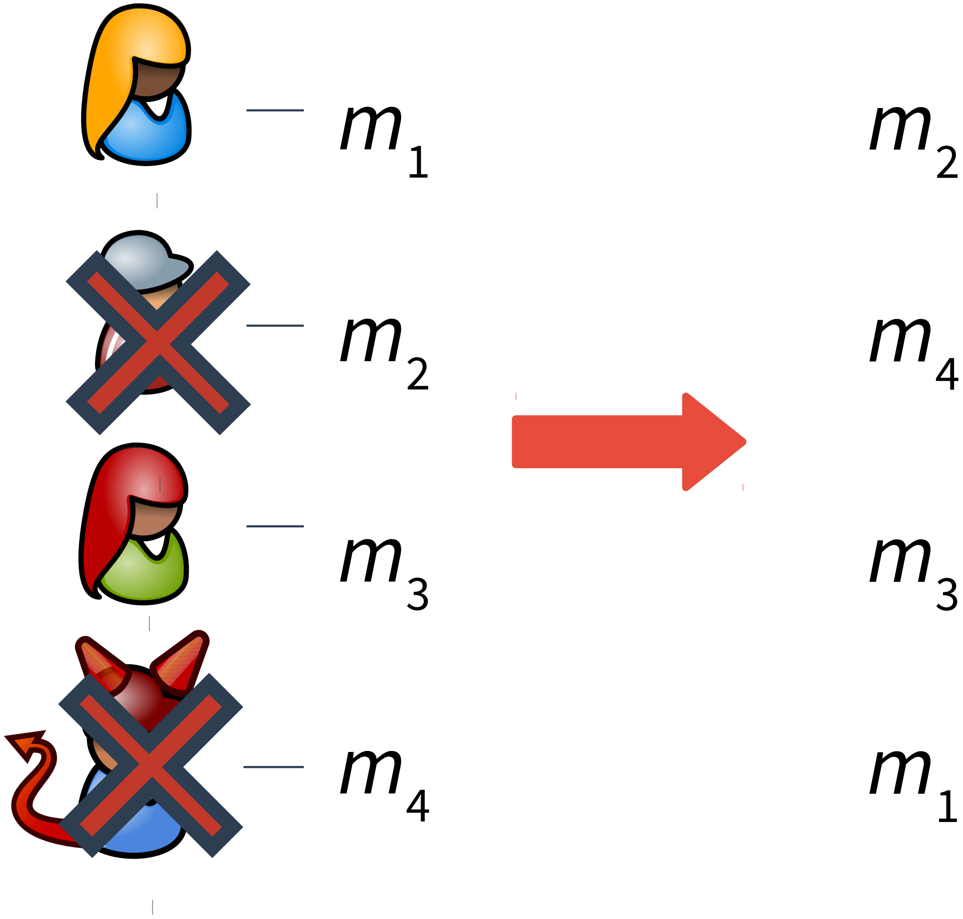
m_4

m_3

m_1

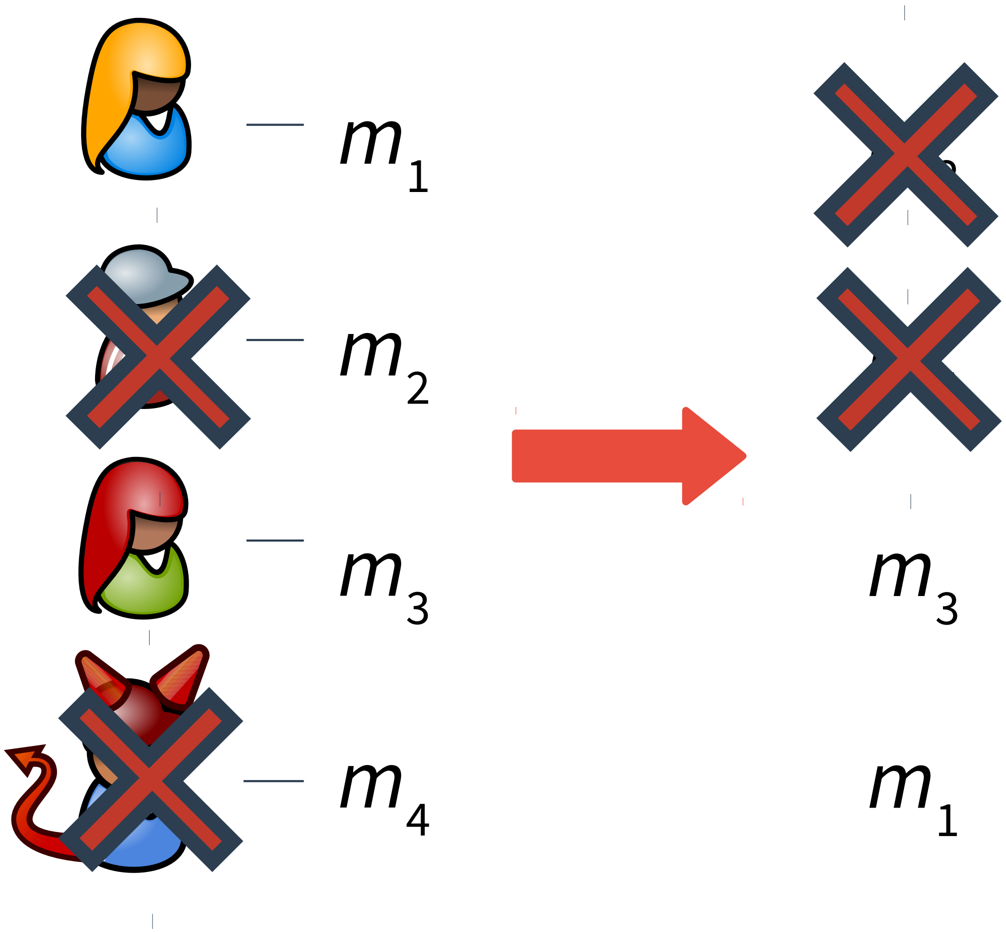
$$M = \{m_1, m_2, m_3, m_4\}$$

Idea of Attack on Anonymity



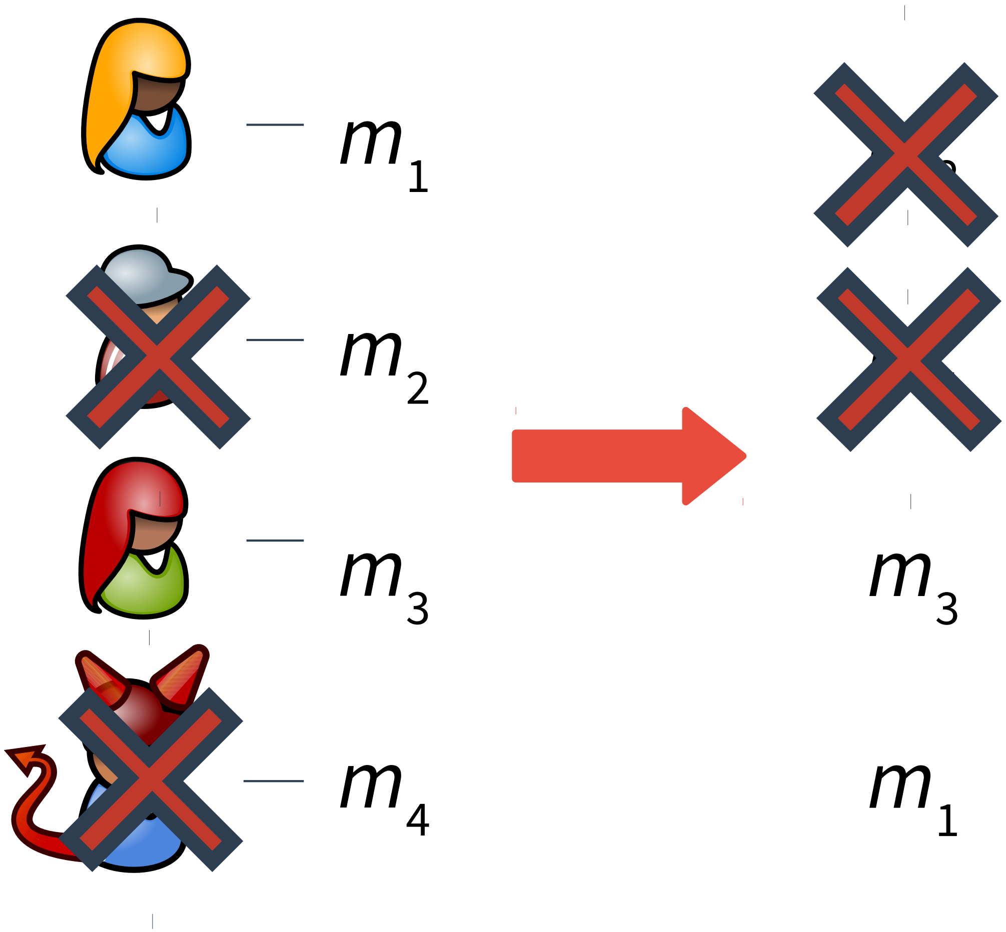
$$M = \{m_1, m_2, m_3, m_4\}$$

Idea of Attack on Anonymity



$$M = \{m_1, m_2, m_3, m_4\}$$

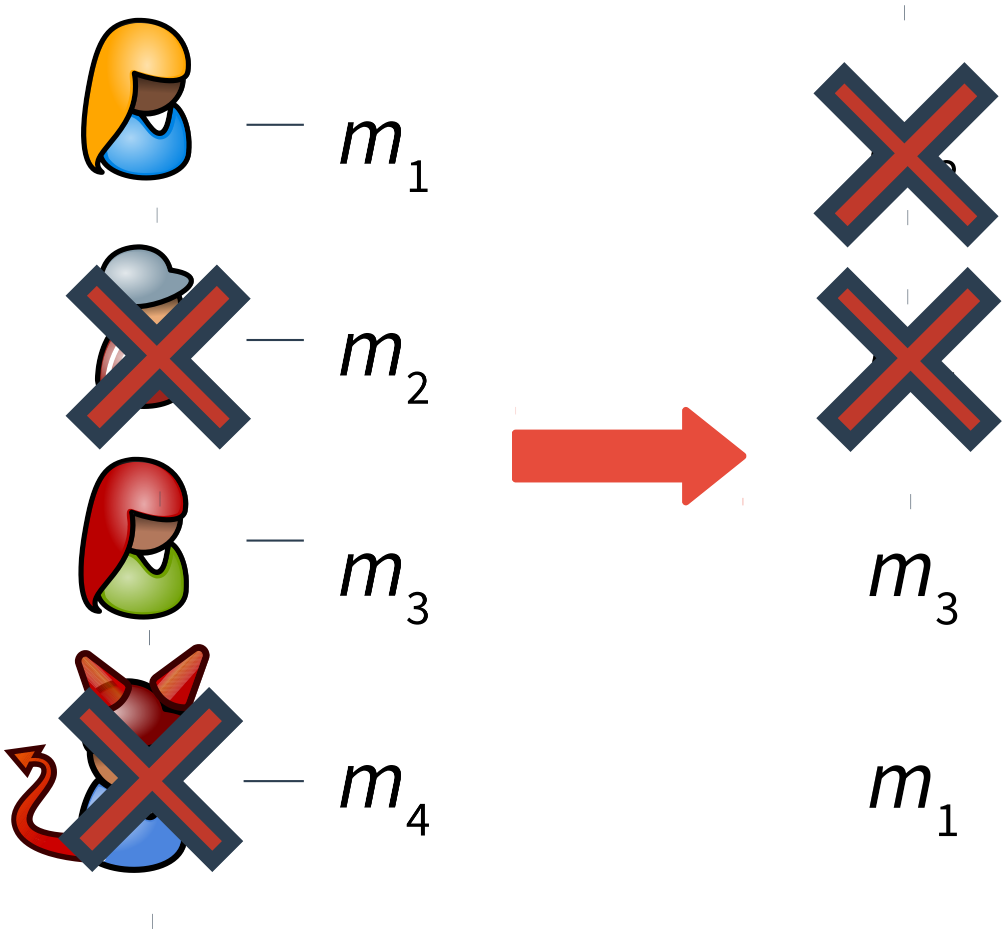
Idea of Attack on Anonymity



$$M = \{m_1, m_2, m_3, m_4\}$$

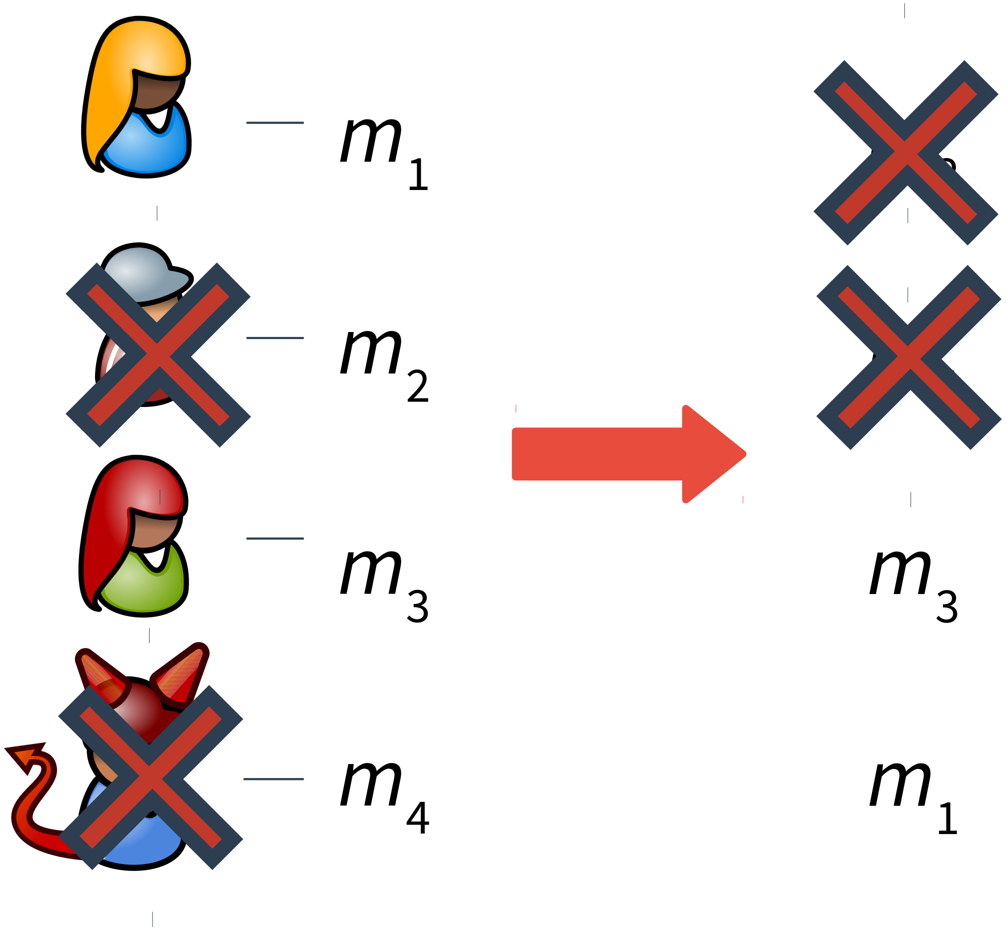
$$M' = \{m_1, m_3\}$$

Idea of Attack on Anonymity



$$M = \{m_1, m_2, m_3, m_4\}$$
$$M' = \{m_1, m_3\}$$
$$M \setminus M' = \{m_2, m_4\}$$

Idea of Attack on Anonymity



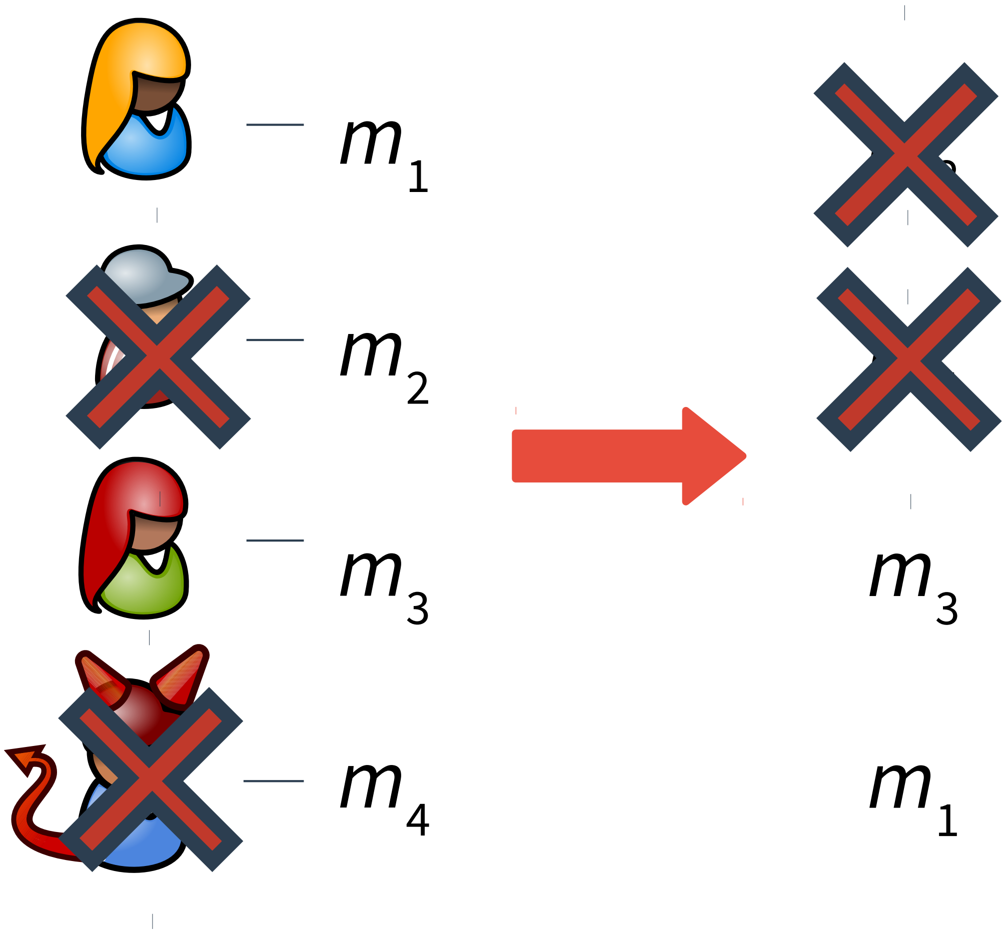
$M = \{m_1, m_2, m_3, m_4\}$

$M' = \{m_1, m_3\}$

$M \setminus M' = \{m_2, m_4\}$

m_4 is attacker’s msg.

Idea of Attack on Anonymity



$M = \{m_1, m_2, m_3, m_4\}$

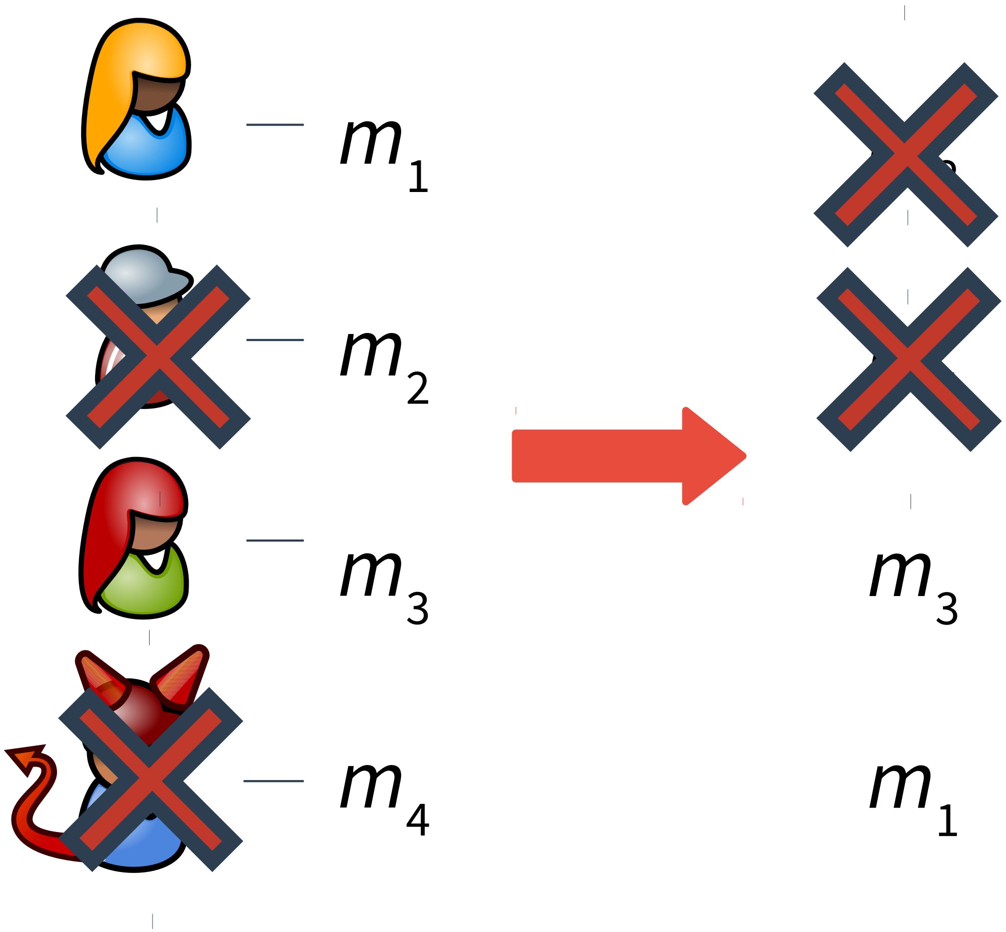
$M' = \{m_1, m_3\}$

$M \setminus M' = \{m_2, m_4\}$

m_4 is attacker’s msg.

m_2 is Bob’s msg.

Idea of Attack on Anonymity



$M = \{m_1, m_2, m_3, m_4\}$

$M' = \{m_1, m_3\}$

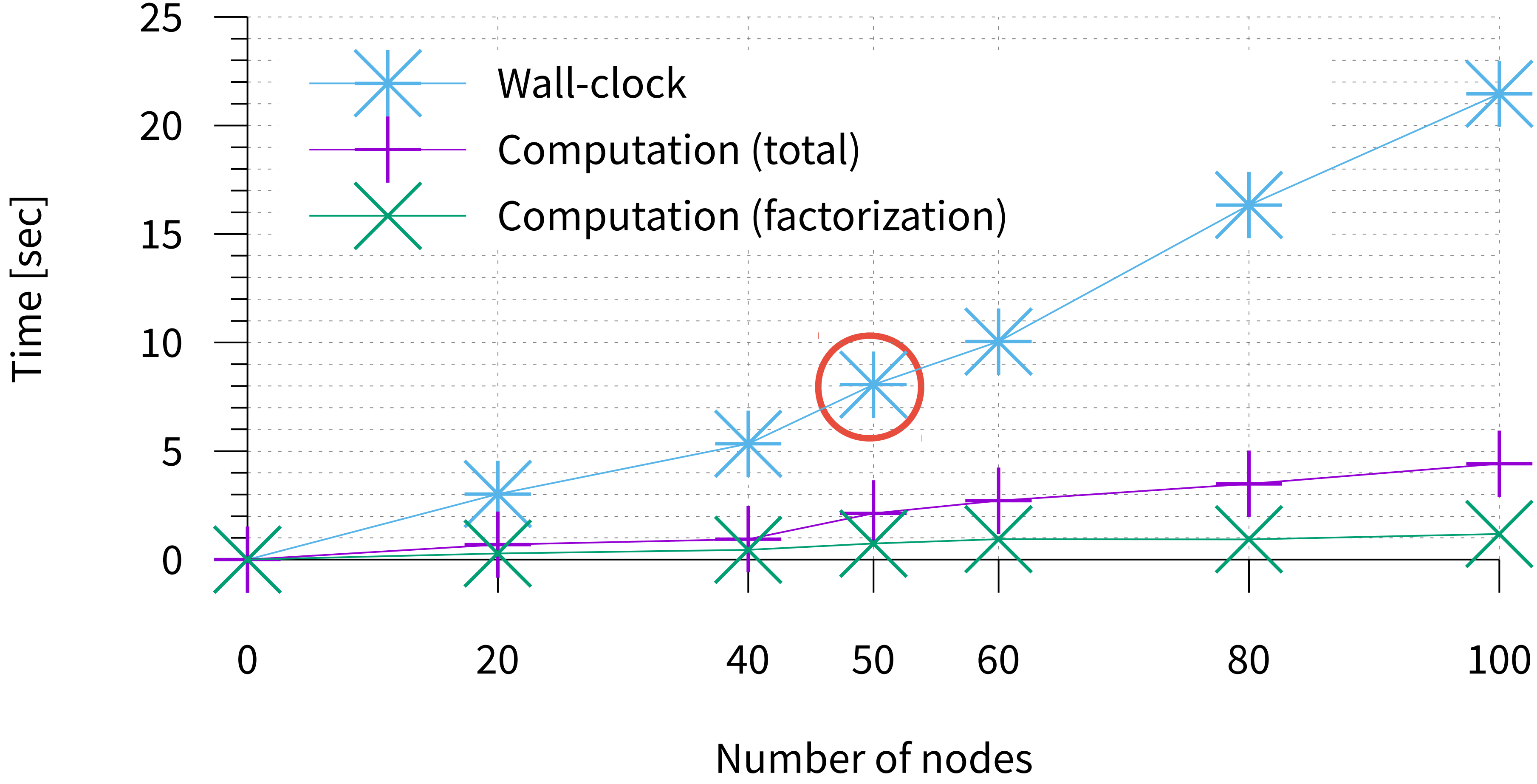
$M \setminus M' = \{m_2, m_4\}$

m_4 is attacker’s msg.

m_2 is Bob’s msg.

Practical attack against Dissent protocol [CCS 2013]!

Performance



DiceMix

A Practical P2P Mixing Protocol based on DC-nets

State of the Art (I)

State of the Art (I)

Mixnet run by all peers

State of the Art (I)

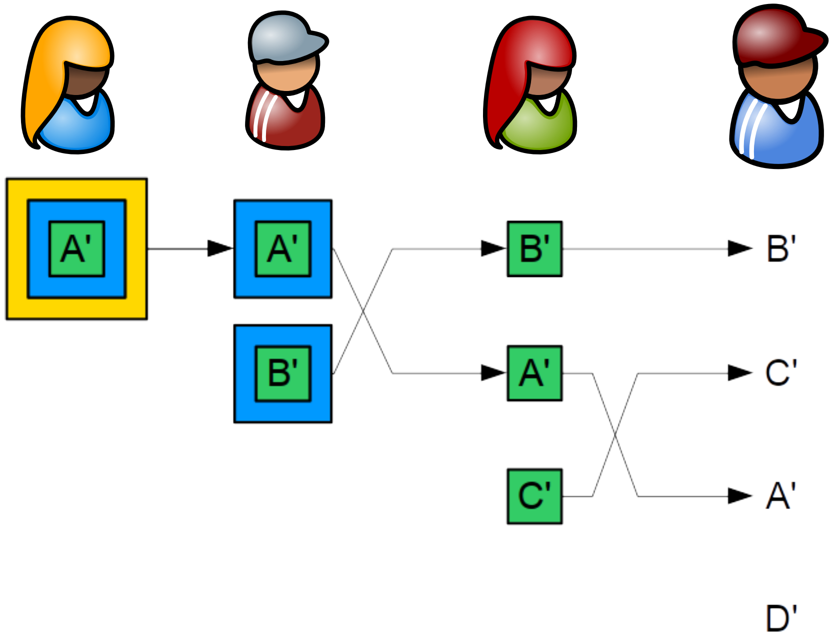
Mixnet run by all peers

- Dissent (shuffle protocol) [CCS 2010], CoinShuffle [ESORICS 2014]

State of the Art (I)

Mixnet run by all peers

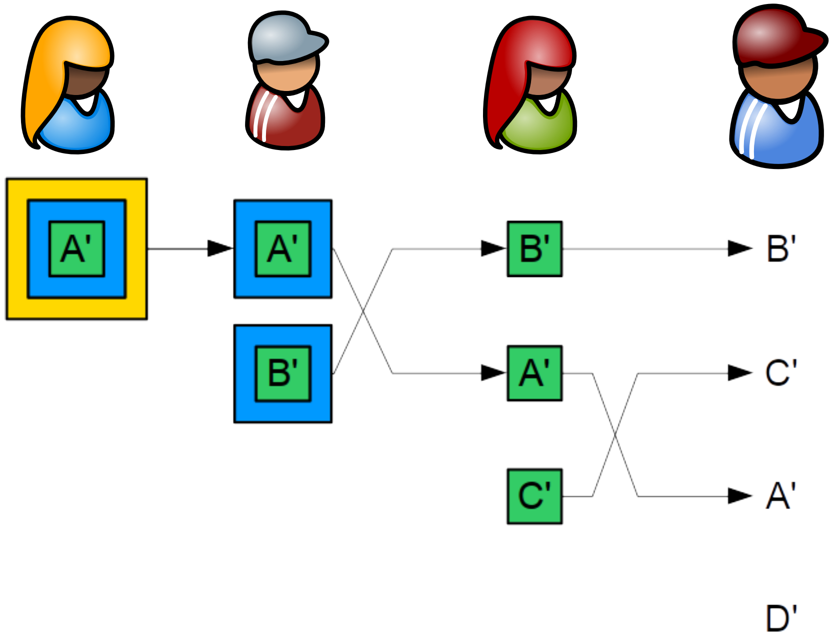
- Dissent (shuffle protocol) [CCS 2010], CoinShuffle [ESORICS 2014]



State of the Art (I)

Mixnet run by all peers

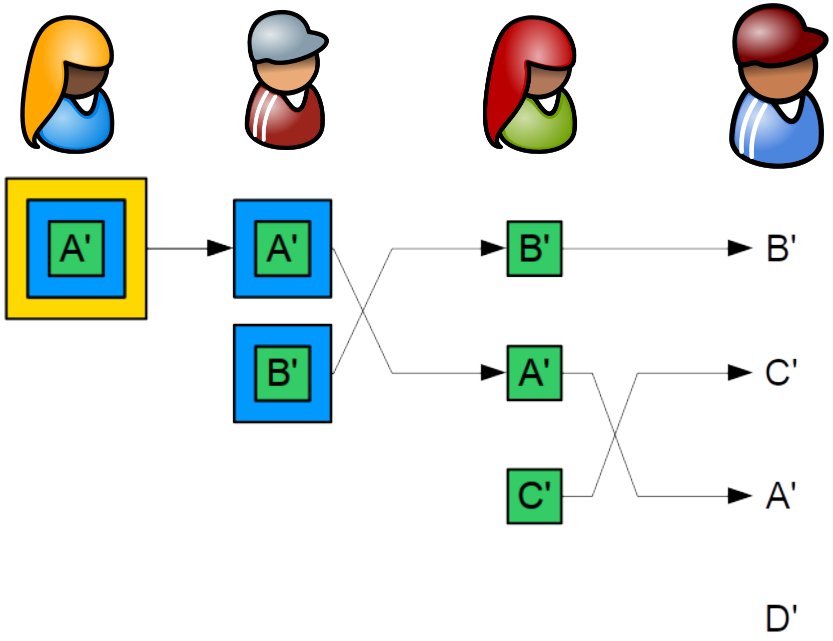
- Dissent (shuffle protocol) [CCS 2010], CoinShuffle [ESORICS 2014]
- O(*n*) rounds in optimistic case



State of the Art (I)

Mixnet run by all peers

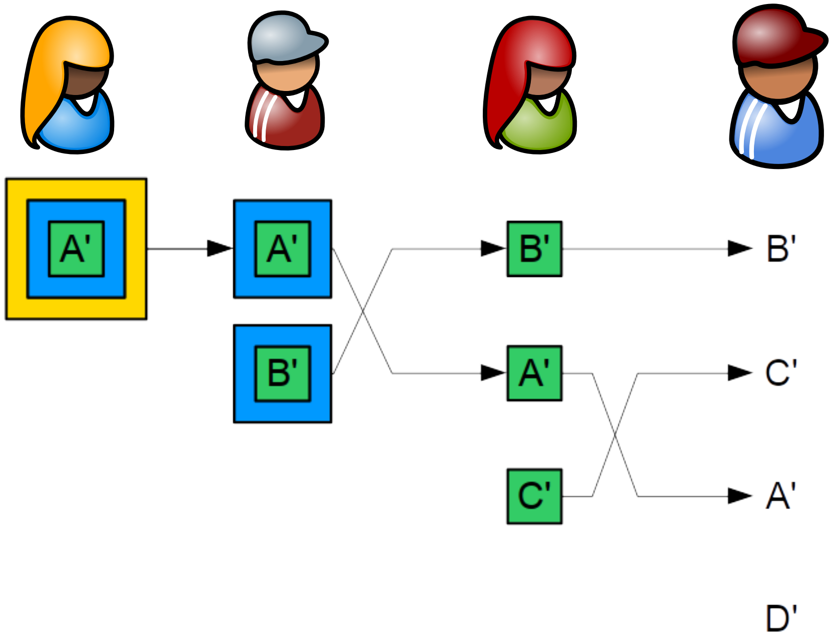
- Dissent (shuffle protocol) [CCS 2010], CoinShuffle [ESORICS 2014]
- O(*n*) rounds in optimistic case
- O(*nf*) rounds for *f* malicious peers



State of the Art (I)

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



Mixnet solution does not scale!

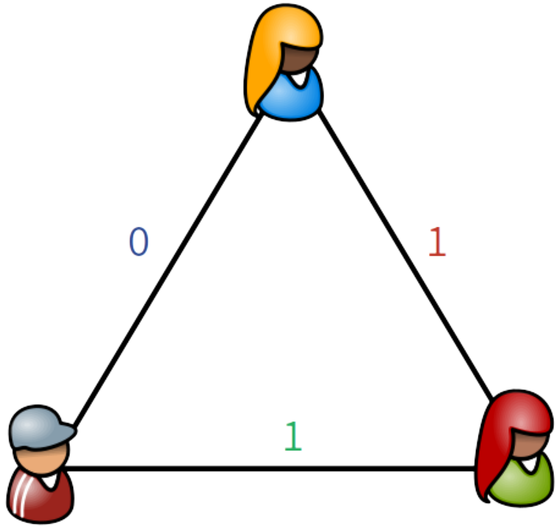
State of the Art (II)

State of the Art (II)

Dining cryptographers' networks (DC-nets)

State of the Art (II)

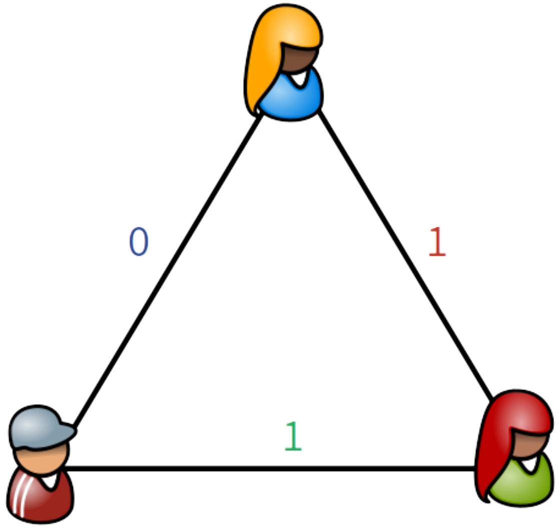
Dining cryptographers' networks (DC-nets)



State of the Art (II)

Dining cryptographers' networks (DC-nets)

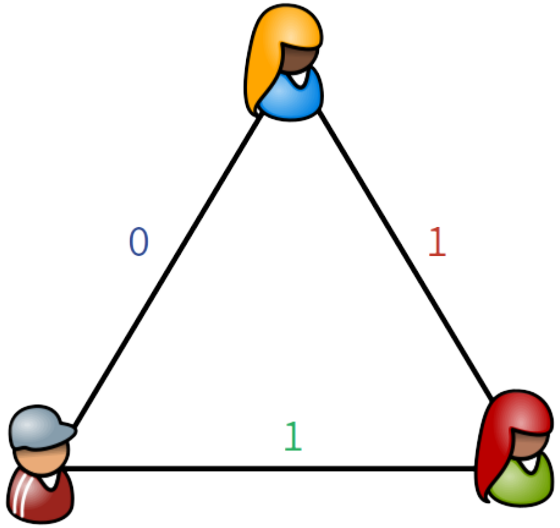
- Hope for O(1) rounds in the optimistic case



State of the Art (II)

Dining cryptographers' networks (DC-nets)

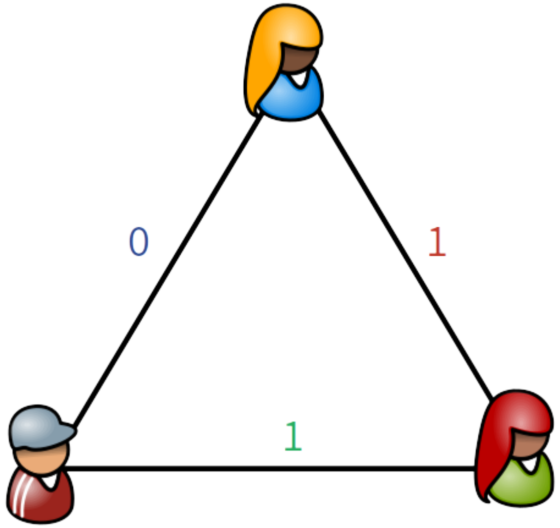
- Hope for O(1) rounds in the optimistic case
- Easy to disrupt



State of the Art (II)

Dining cryptographers' networks (DC-nets)

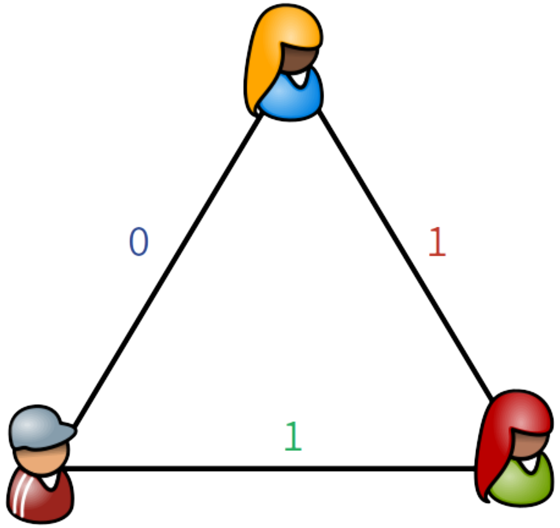
- Hope for O(1) rounds in the optimistic case
- Easy to disrupt
- All approaches to solve disruption problem suffer from drawbacks



State of the Art (II)

Dining cryptographers' networks (DC-nets)

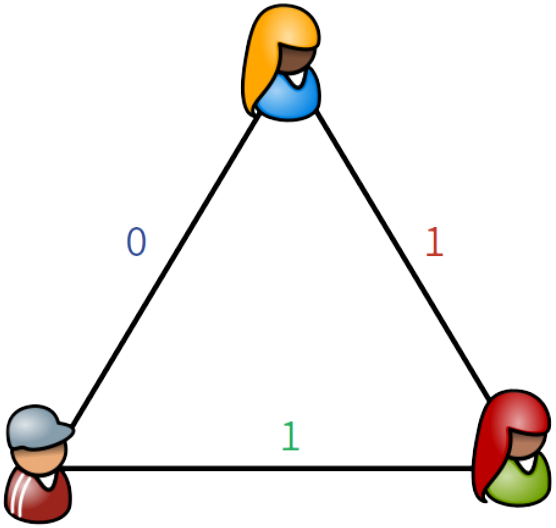
- Hope for O(1) rounds in the optimistic case
- Easy to disrupt
- All approaches to solve disruption problem suffer from drawbacks
- Golle and Juels [EUROCRYPT 2004]: Honest majority



State of the Art (II)

Dining cryptographers' networks (DC-nets)

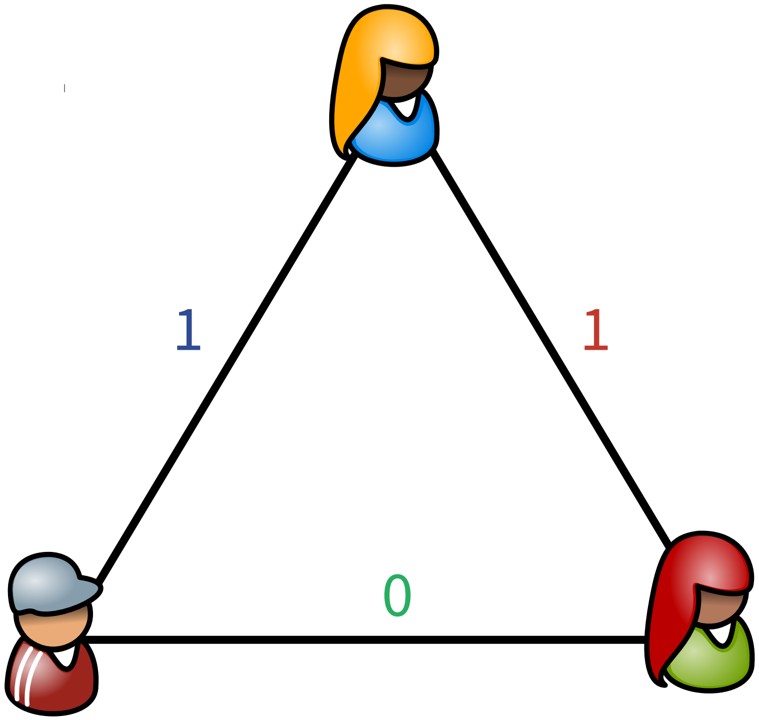
- Hope for O(1) rounds in the optimistic case
- Easy to disrupt
- All approaches to solve disruption problem suffer from drawbacks
- Golle and Juels [EUROCRYPT 2004]: Honest majority



No practical P2P mixing protocol based on DC-nets!

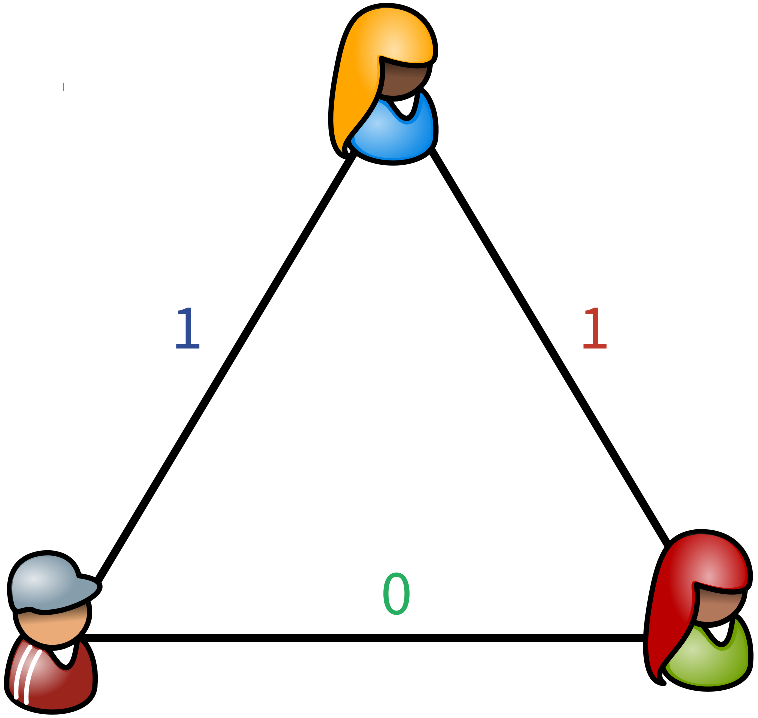
DC-net [Chaum 1988]

DC-net [Chaum 1988]

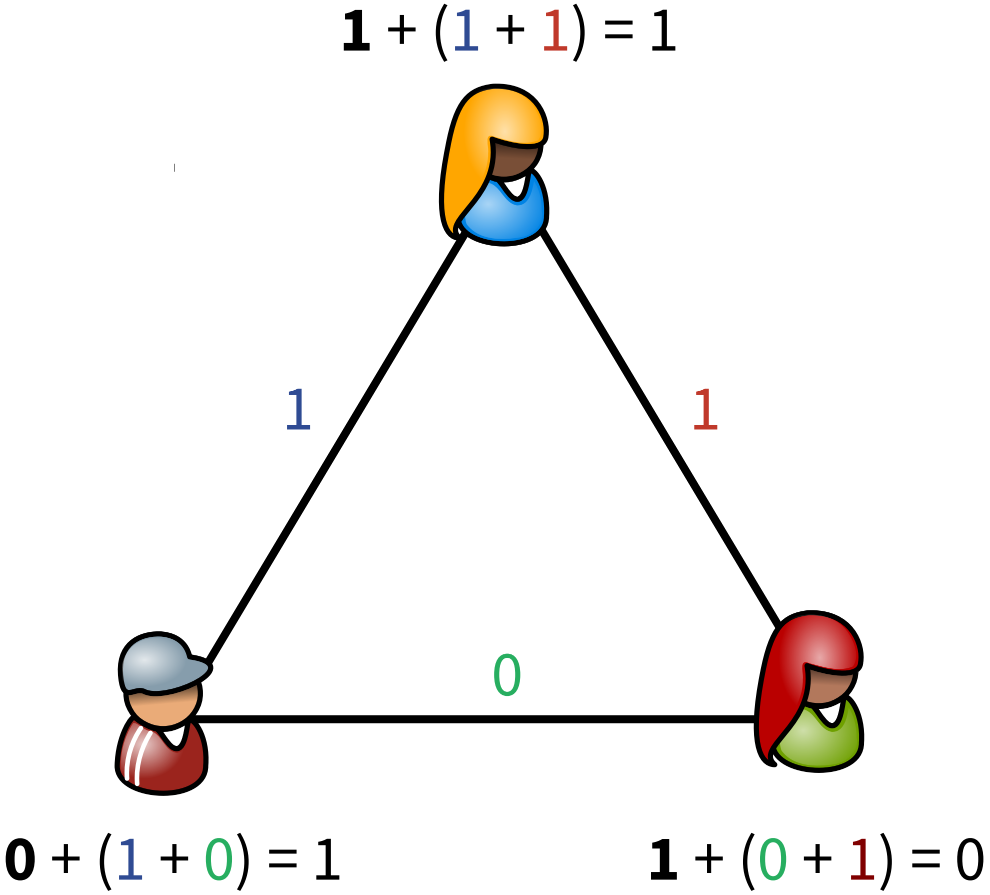


DC-net [Chaum 1988]

$\mathbf{1} + ({\color{blue}1} + {\color{red}1}) = 1$



DC-net [Chaum 1988]



$\mathbf{1} + (1 + 1) = 1$

1

1

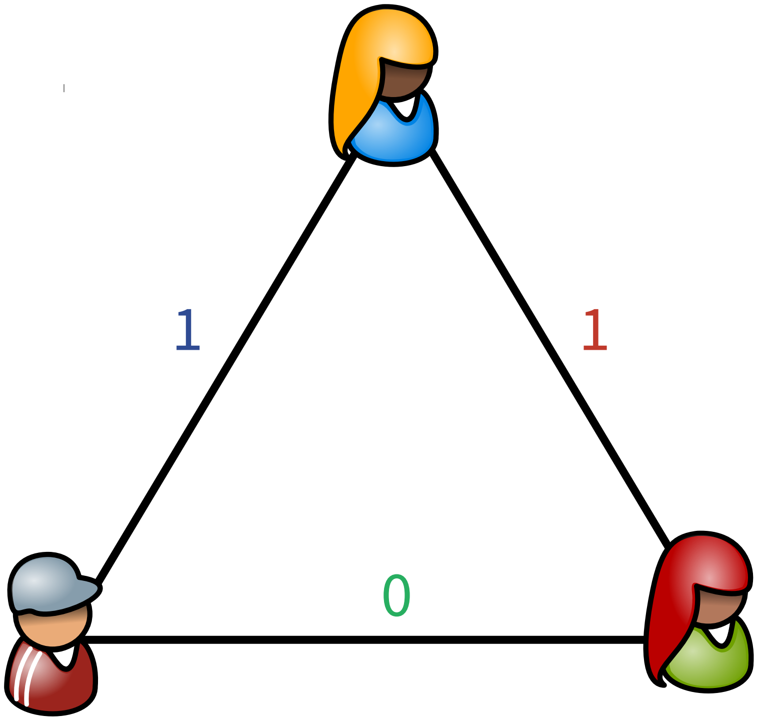
0

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

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

DC-net [Chaum 1988]

$\mathbf{1} + (1 + 1) = 1$



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

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

$1 + 1 + 0 = 0$

Sending Several Messages [Bos, Boer 1989]

User 1:	m_1
User 2:	m_2
User 3:	m_3
	$\vdots$
User *n*:	m_n

$$\sum_{i=1}^{n} m_i$$

Sending Several Messages [Bos, Boer 1989]

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

Sending Several Messages [Bos, Boer 1989]

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

Newton's identities tell us the coefficients of the polynomial $\prod_{i=1}^{n} (x - m_i)$.

Sending Several Messages [Bos, Boer 1989]

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

Newton's identities tell us the coefficients of the polynomial $\prod_{i=1}^{n} (x - m_i)$.

→ Polynomial factorization recovers the messages.

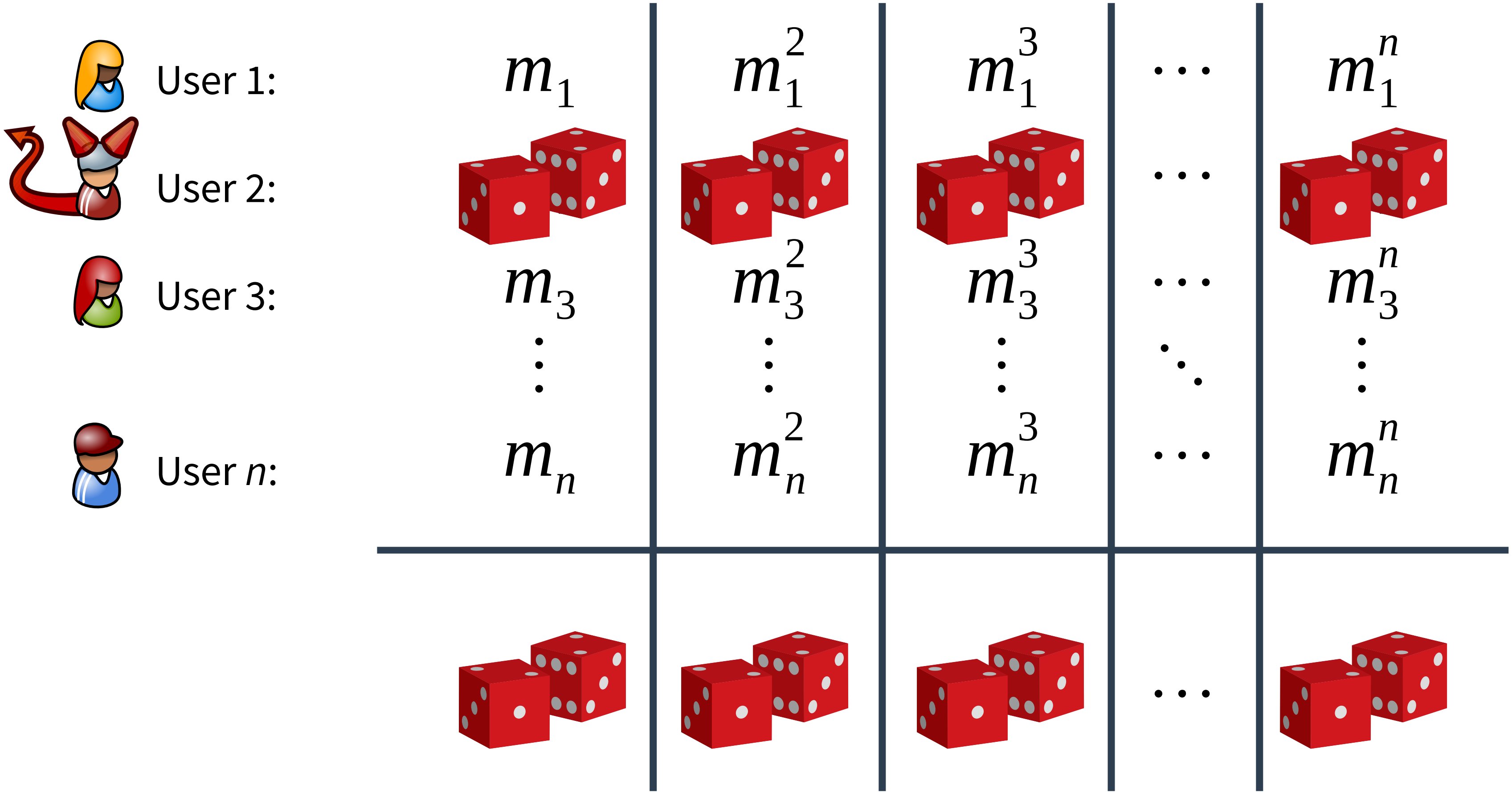
Disruption

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

Disruption

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

Disruption



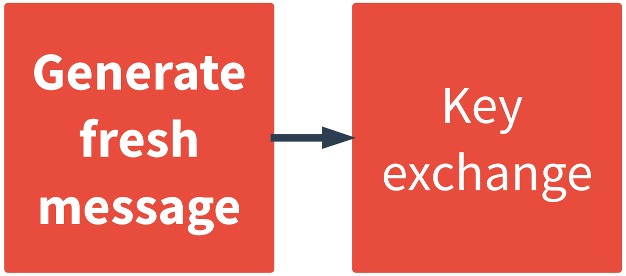
User 1:	m_1	m_1^2	m_1^3	$\cdots$	m_1^n
User 2:	🎲	🎲	🎲	$\cdots$	🎲
User 3:	m_3	m_3^2	m_3^3	$\cdots$	m_3^n
	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
User *n*:	m_n	m_n^2	m_n^3	$\cdots$	m_n^n
	🎲	🎲	🎲	$\cdots$	🎲

Malicious user stays anonymous!

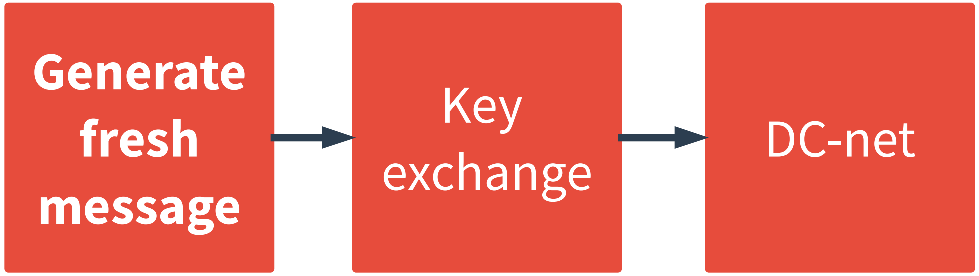
Flowchart of DiceMix

Generate fresh message

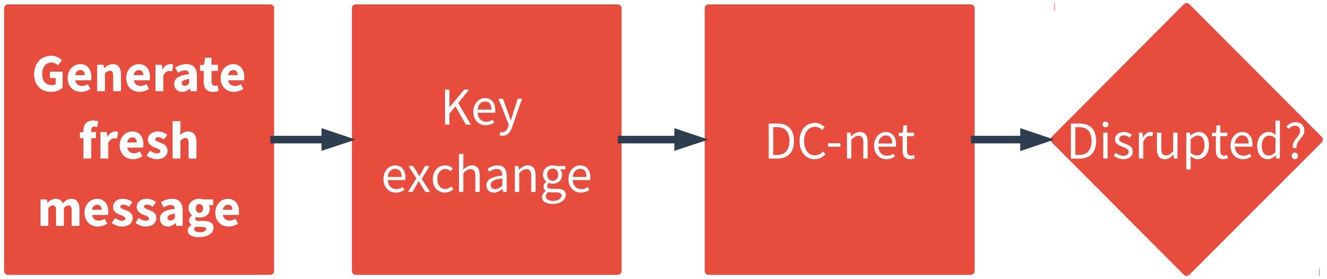
Flowchart of DiceMix



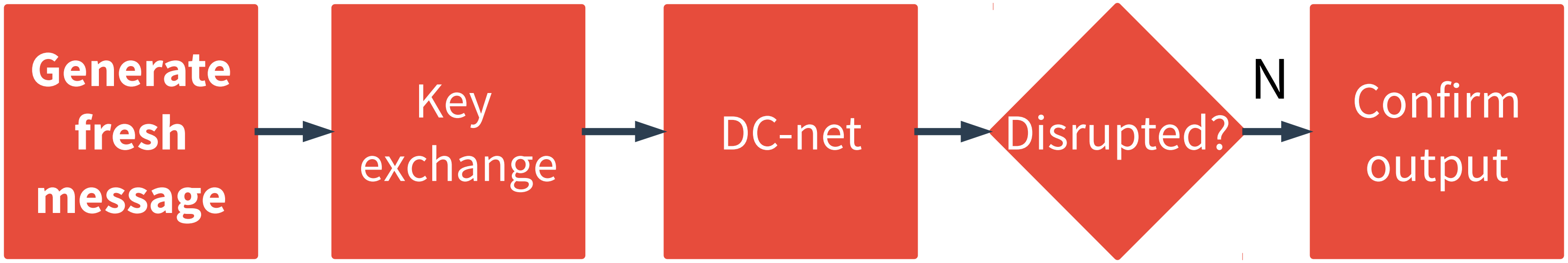
Flowchart of DiceMix



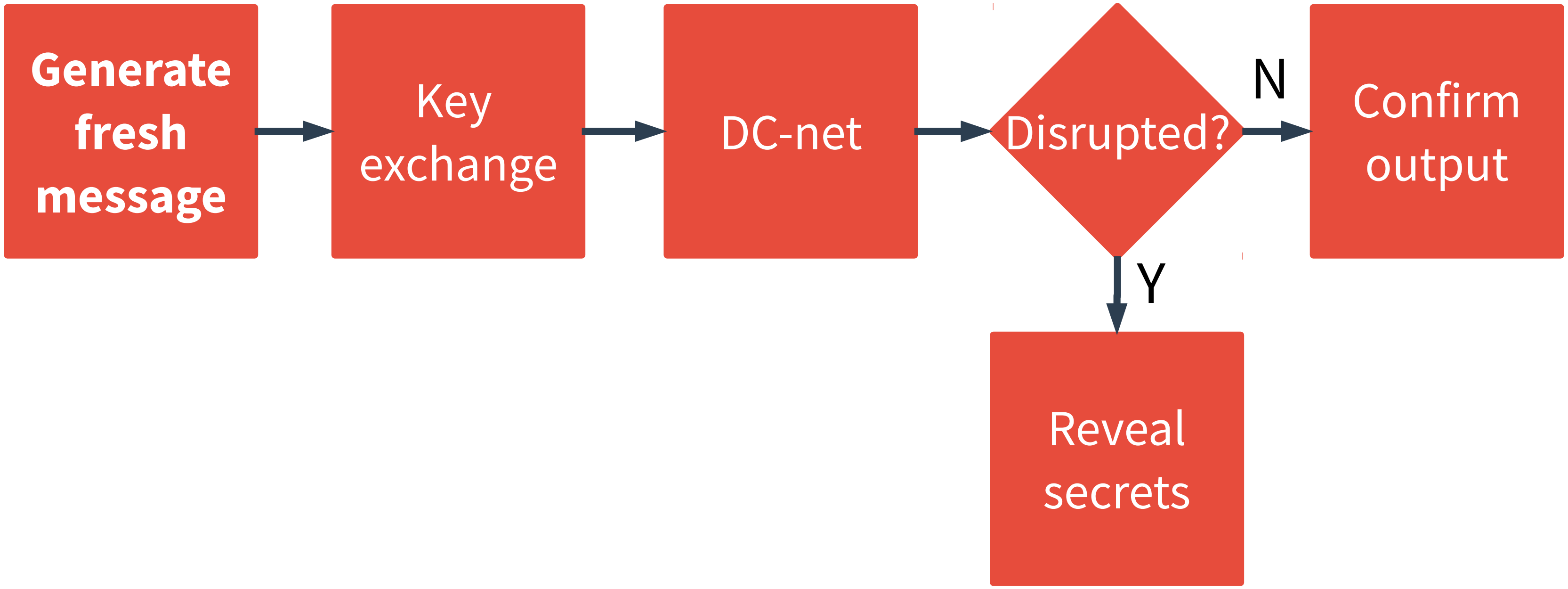
Flowchart of DiceMix



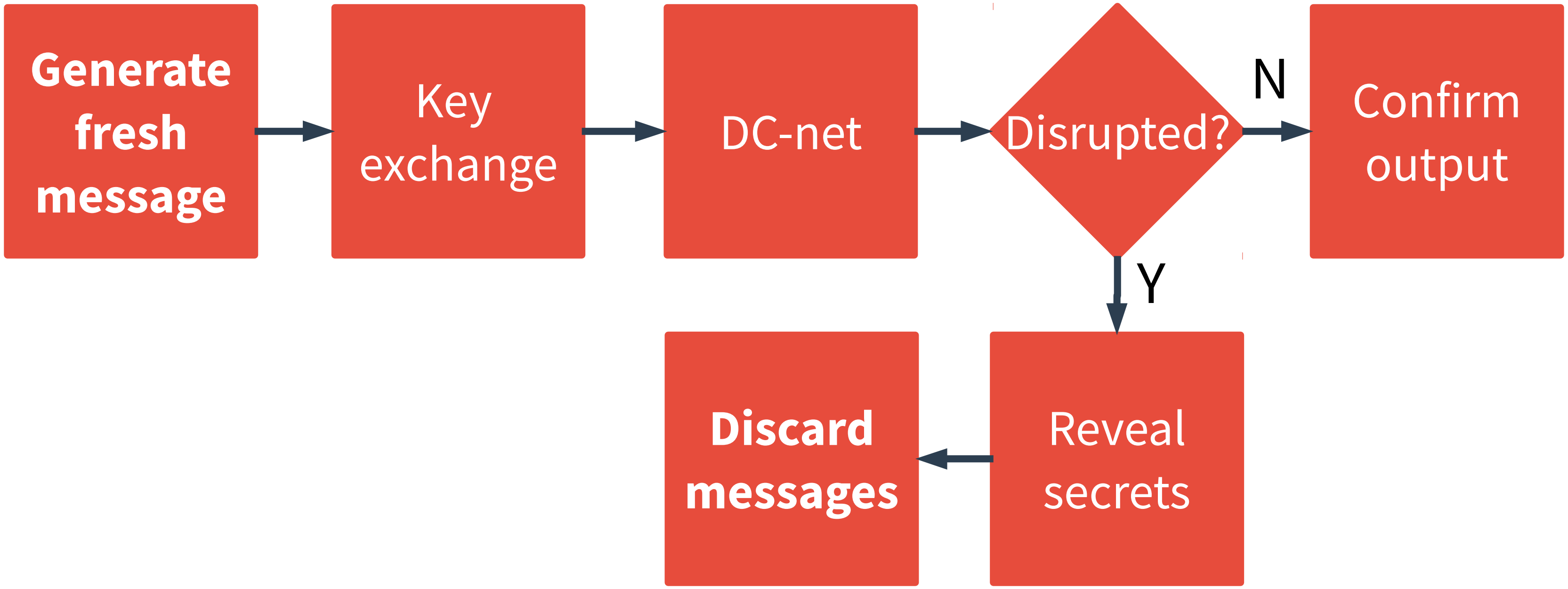
Flowchart of DiceMix



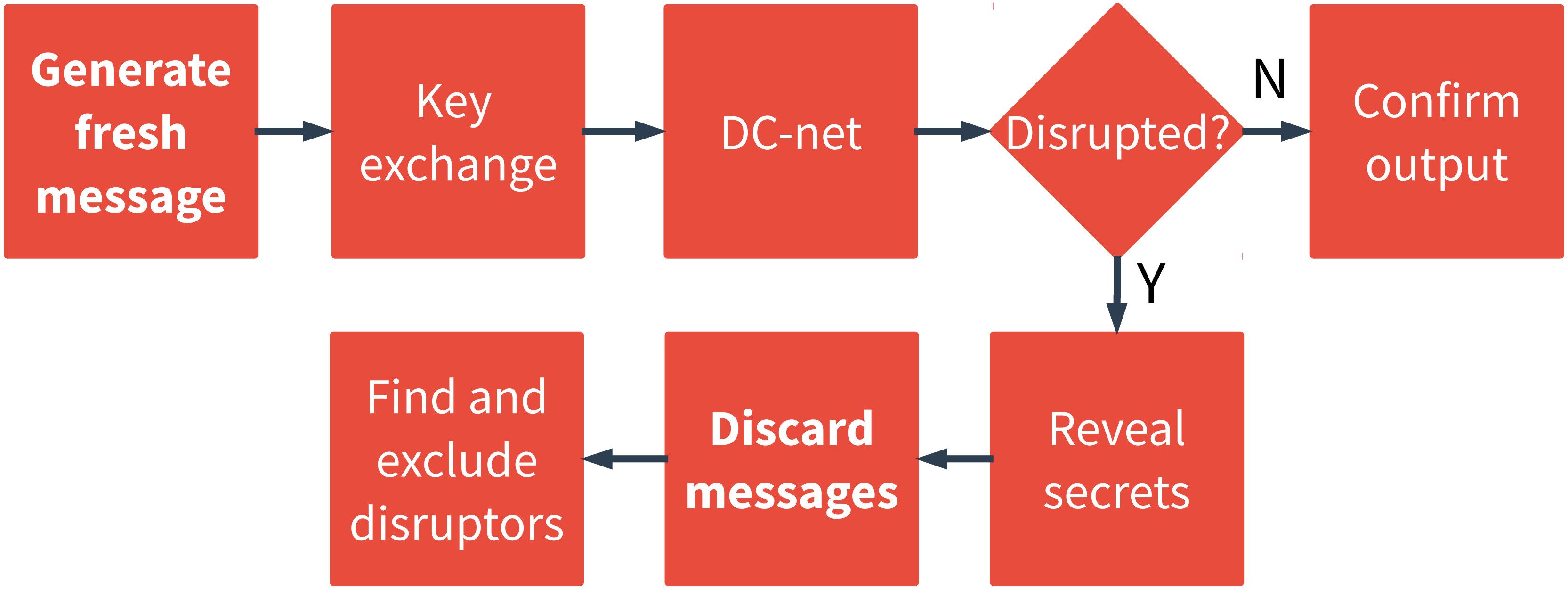
Flowchart of DiceMix



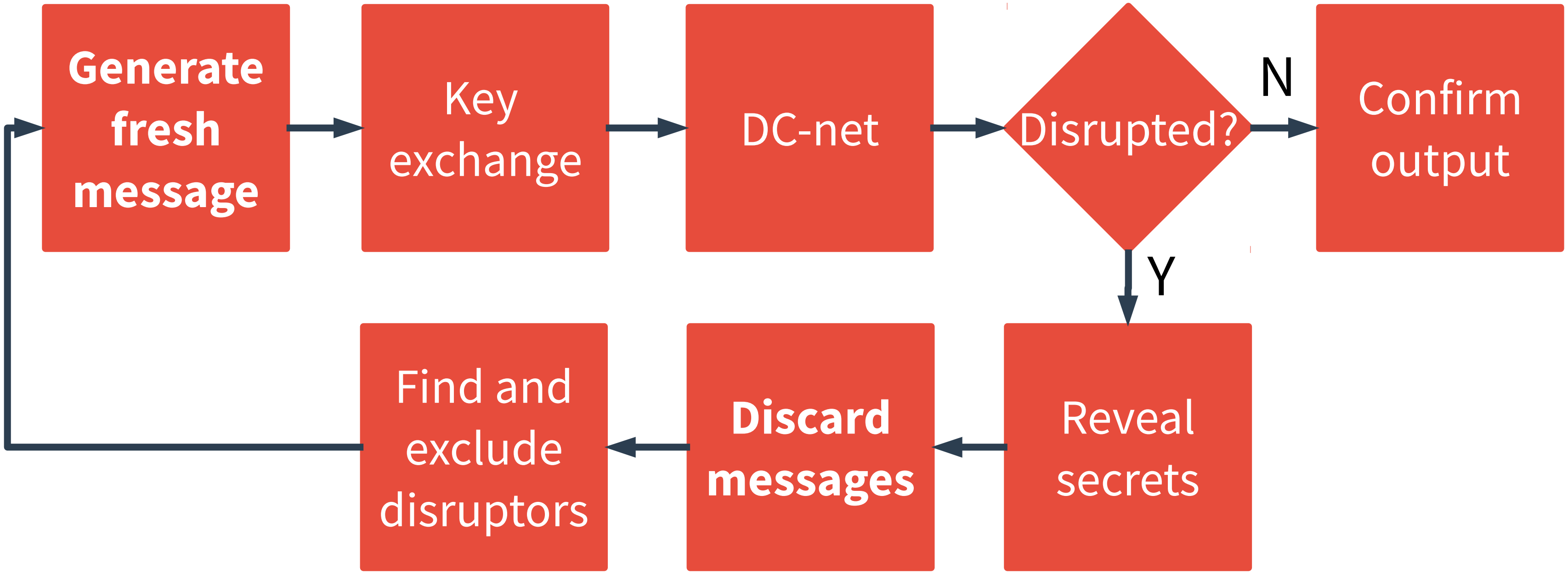
Flowchart of DiceMix



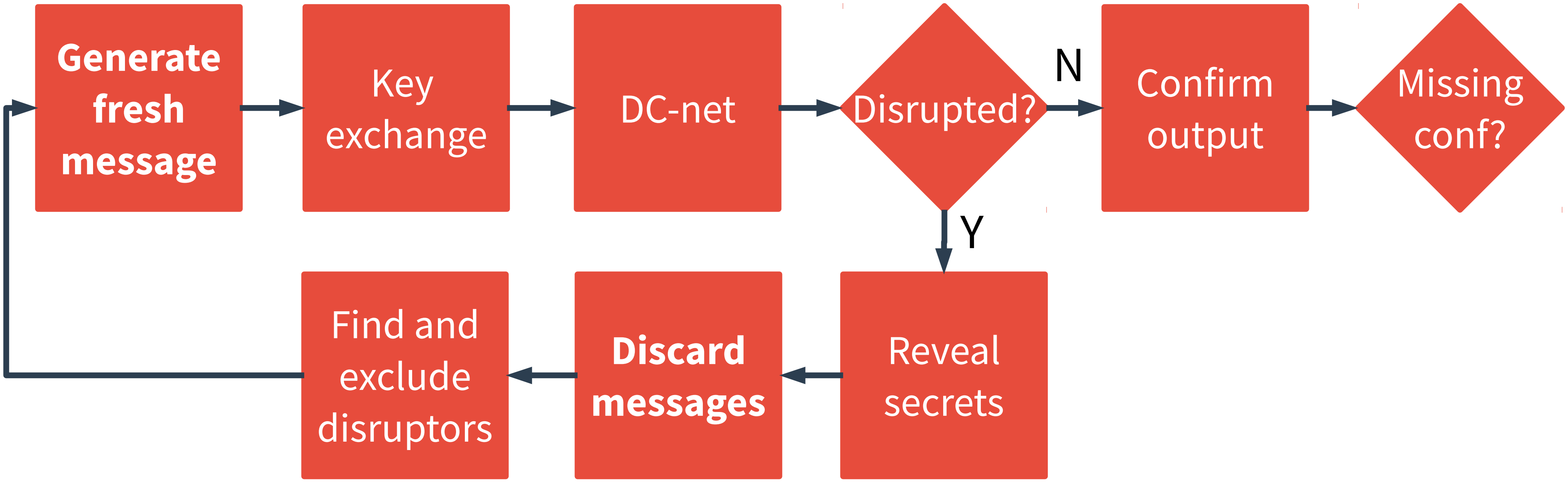
Flowchart of DiceMix



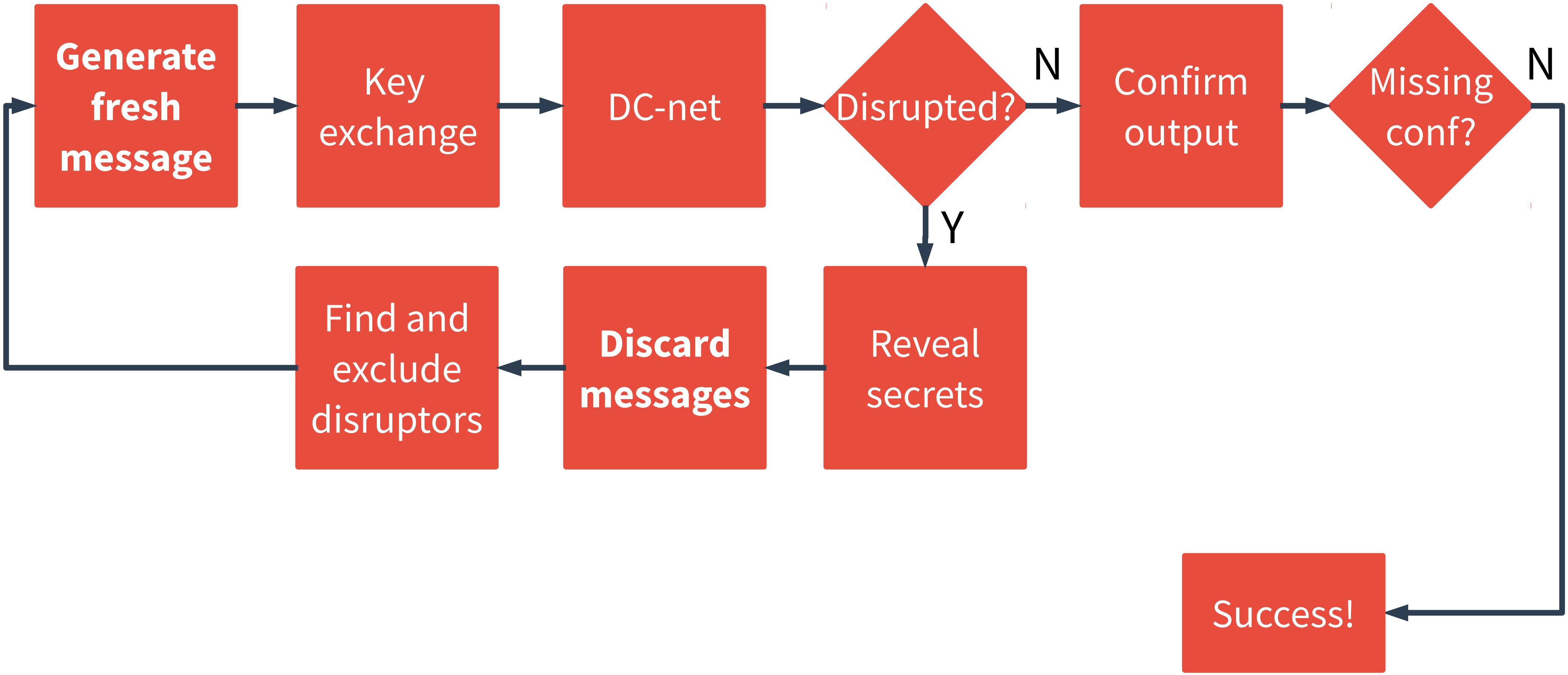
Flowchart of DiceMix



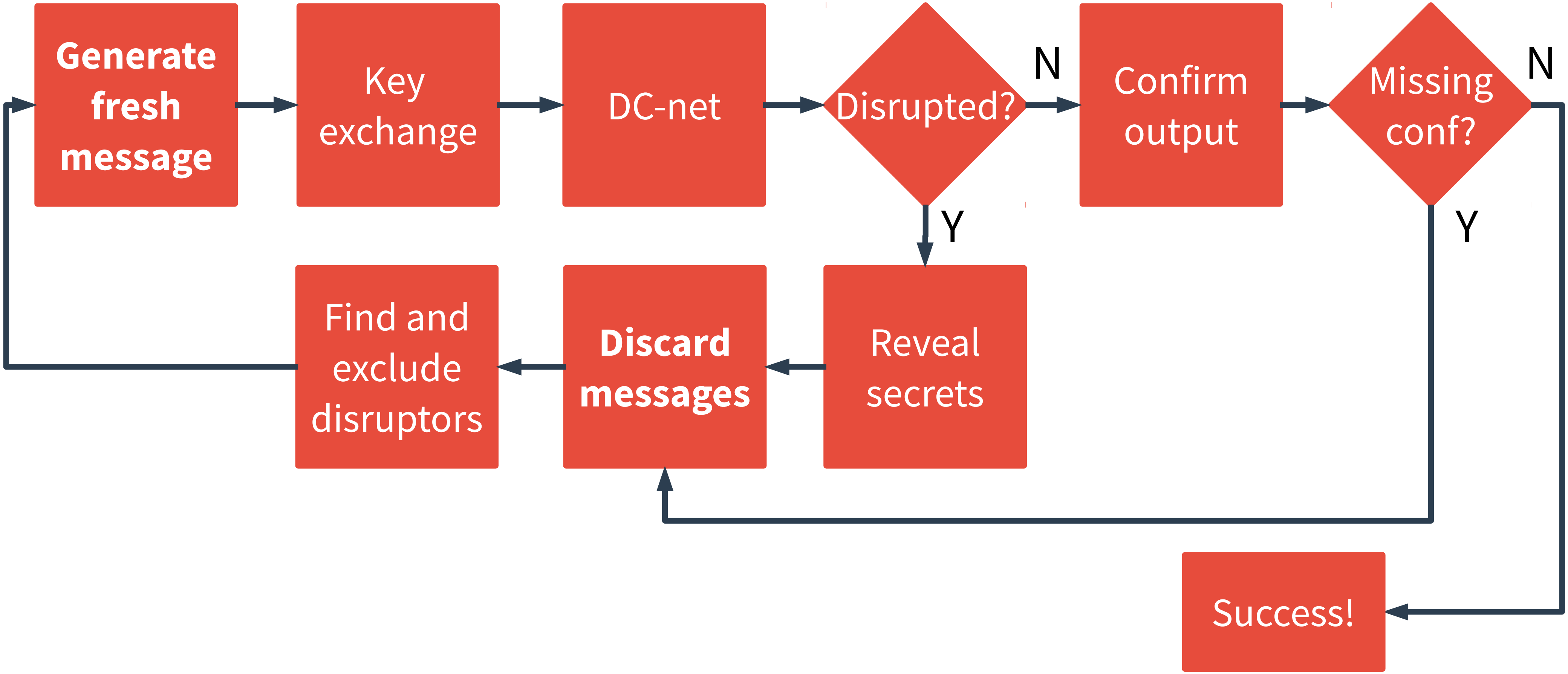
Flowchart of DiceMix



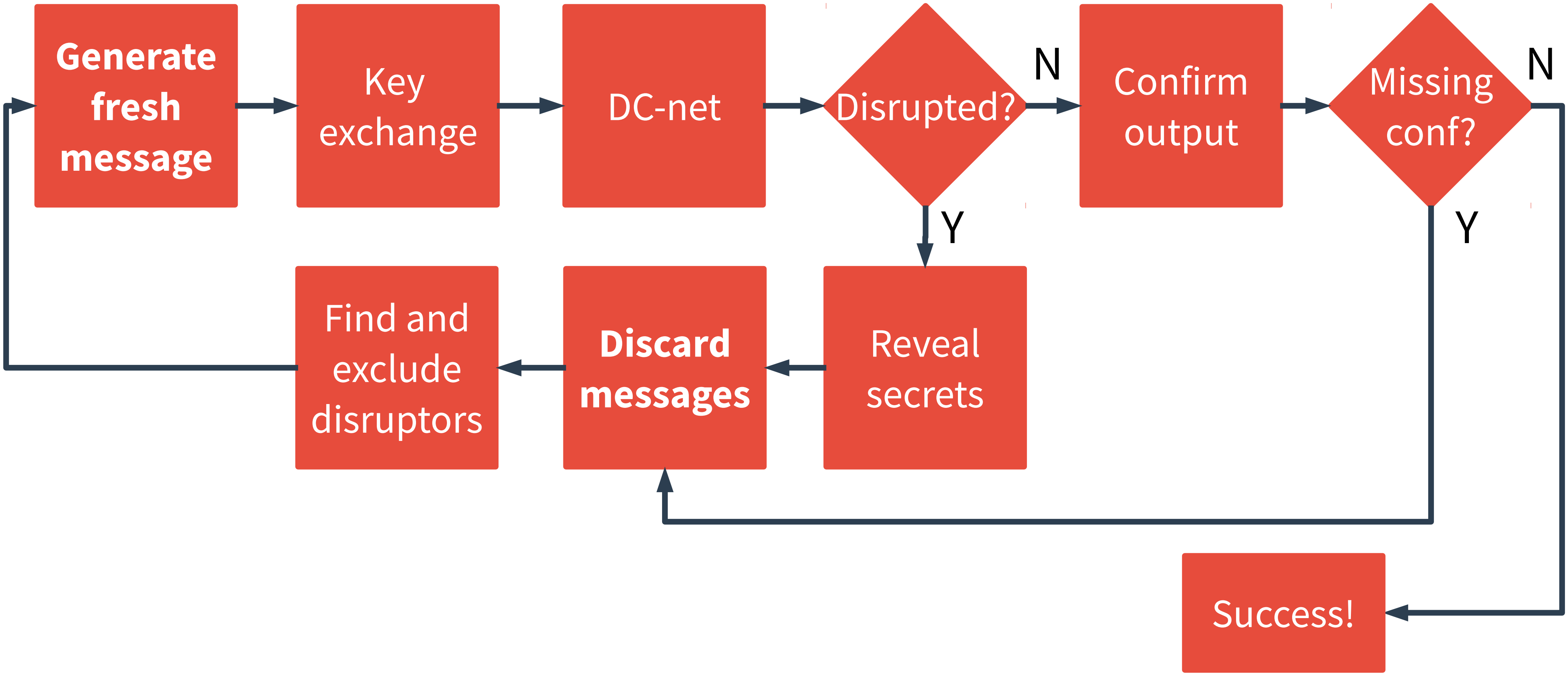
Flowchart of DiceMix



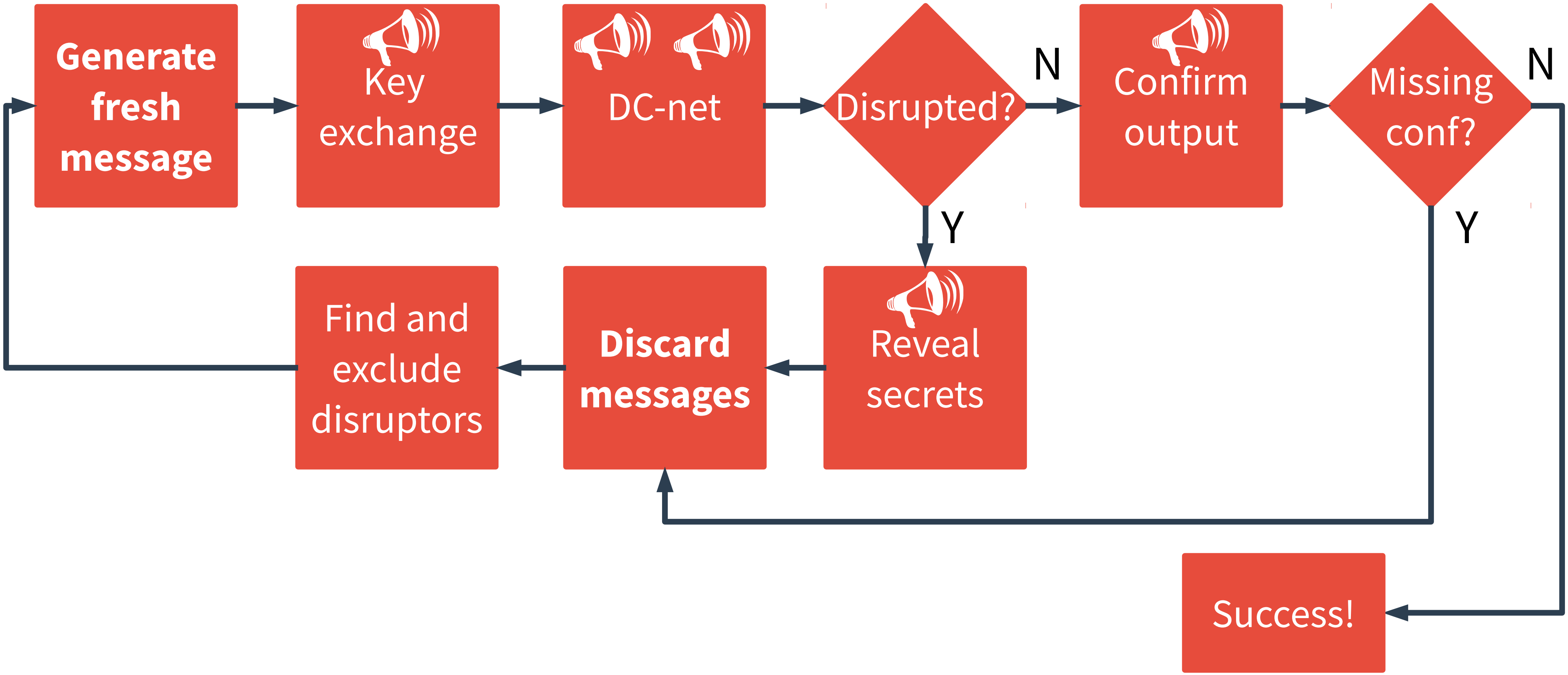
Flowchart of DiceMix



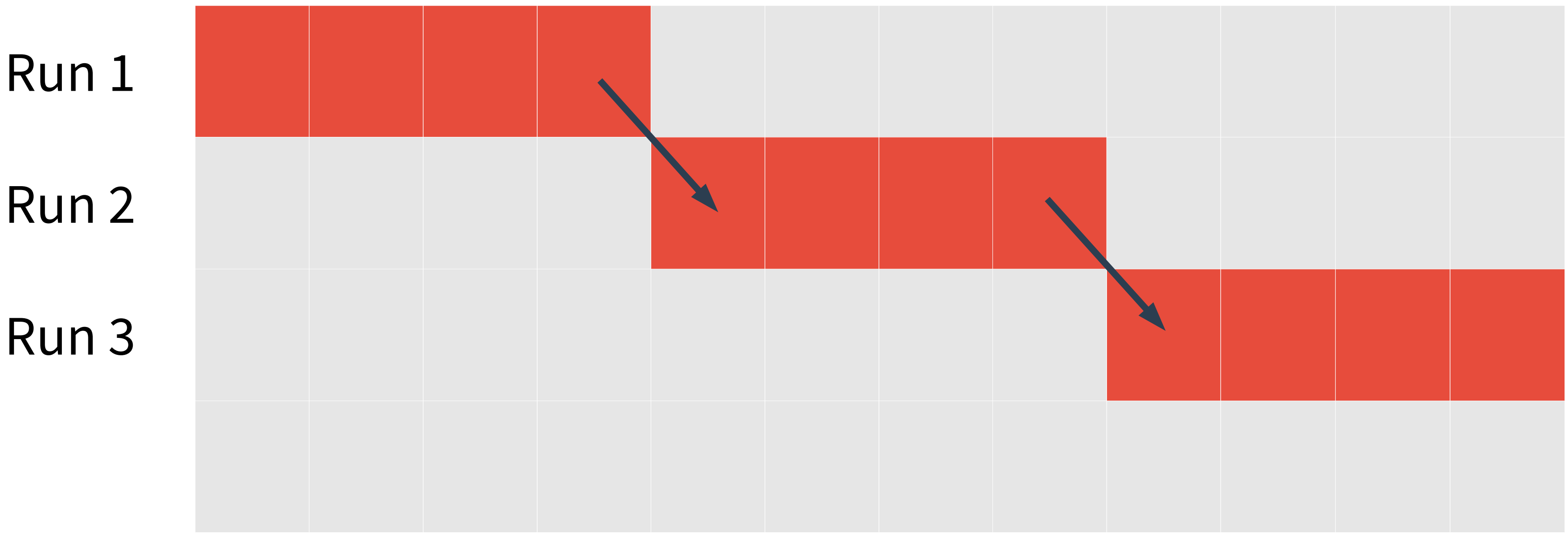
Flowchart of DiceMix



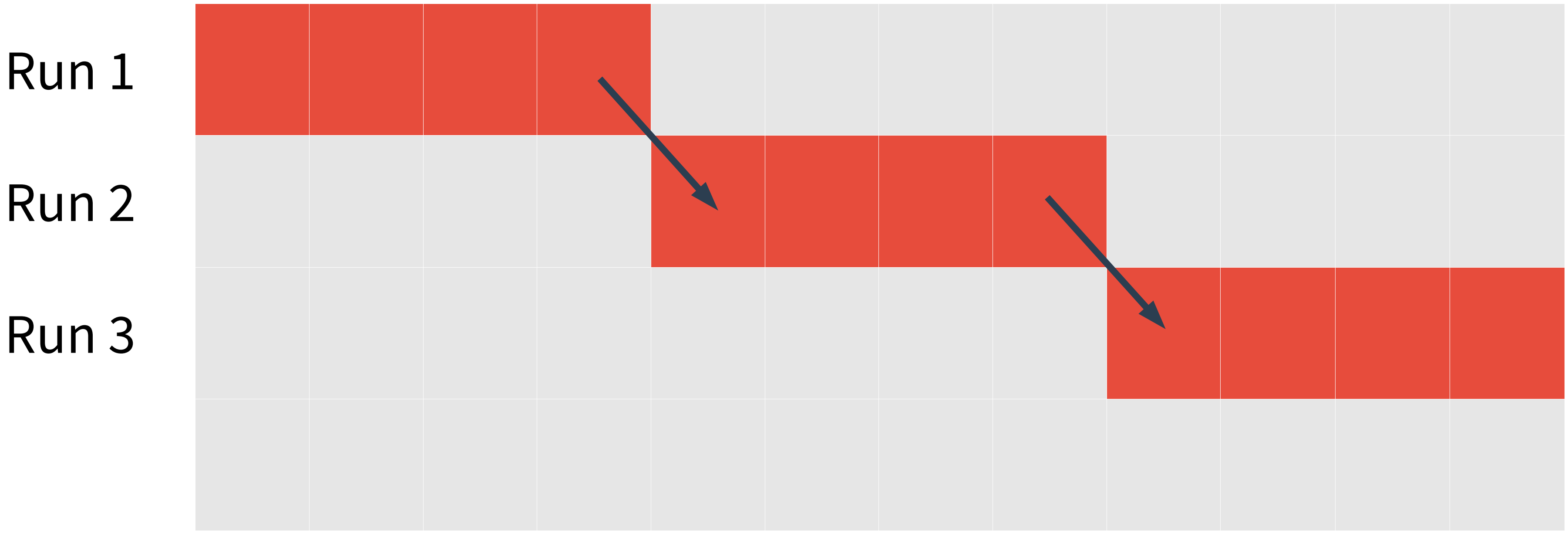
Flowchart of DiceMix



Communication Rounds (naive)



Communication Rounds (naive)

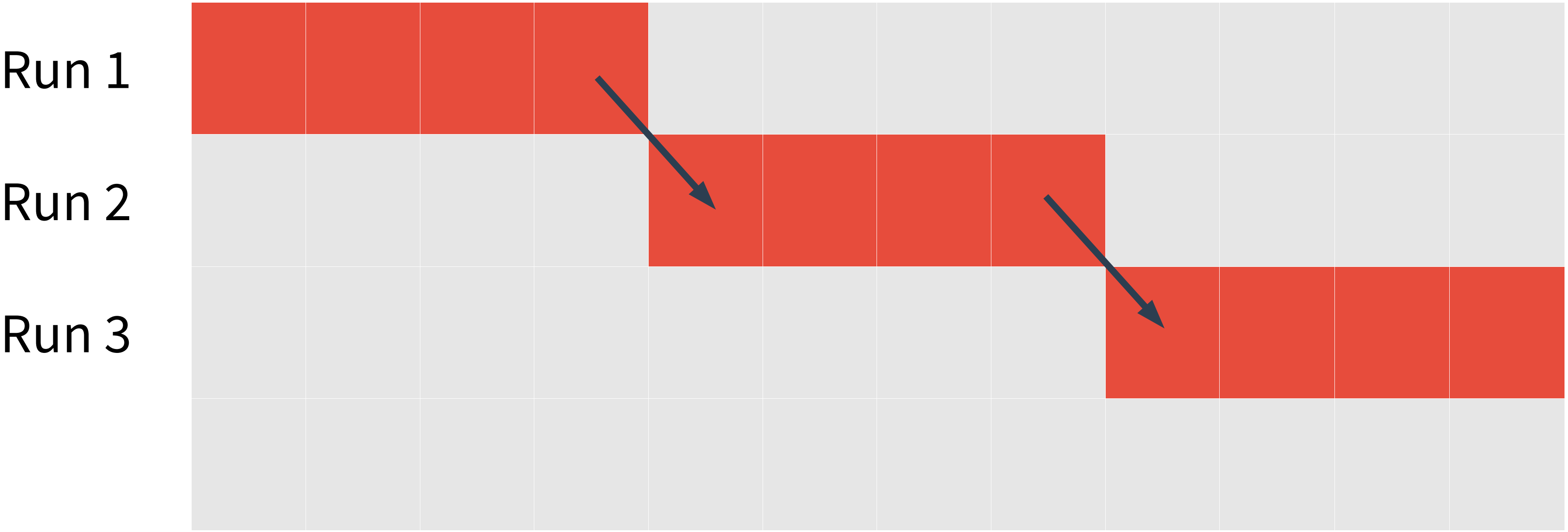


Run 1

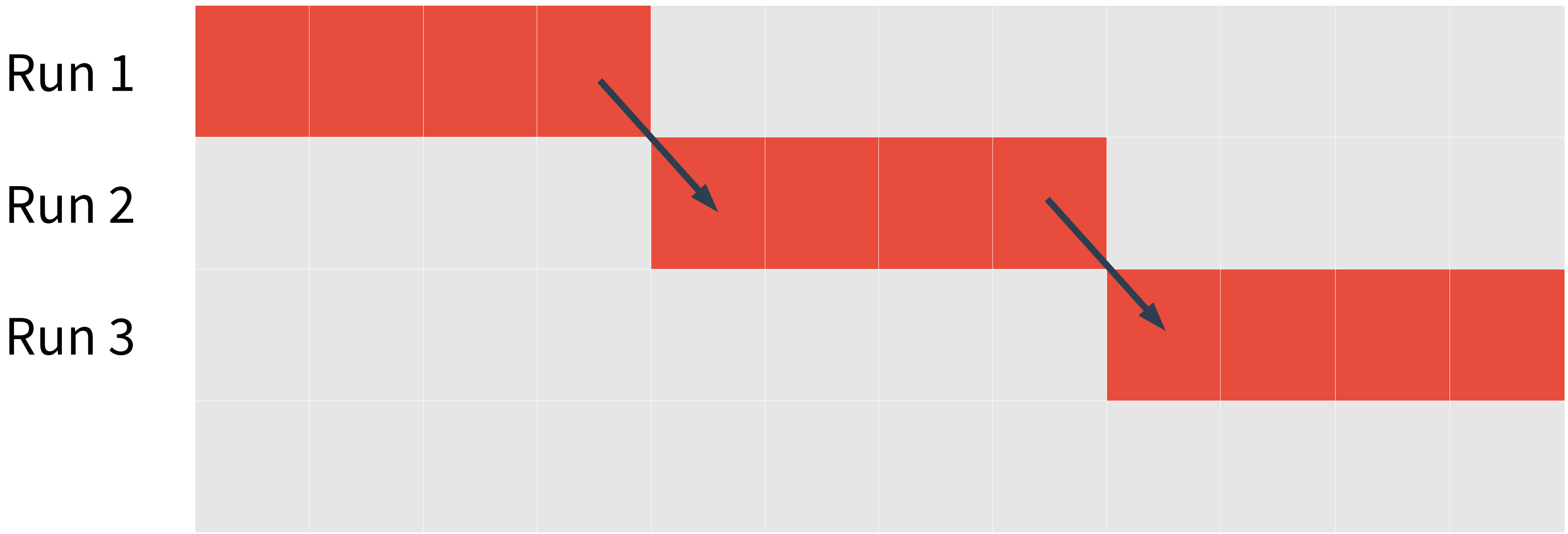
Run 2

Run 3

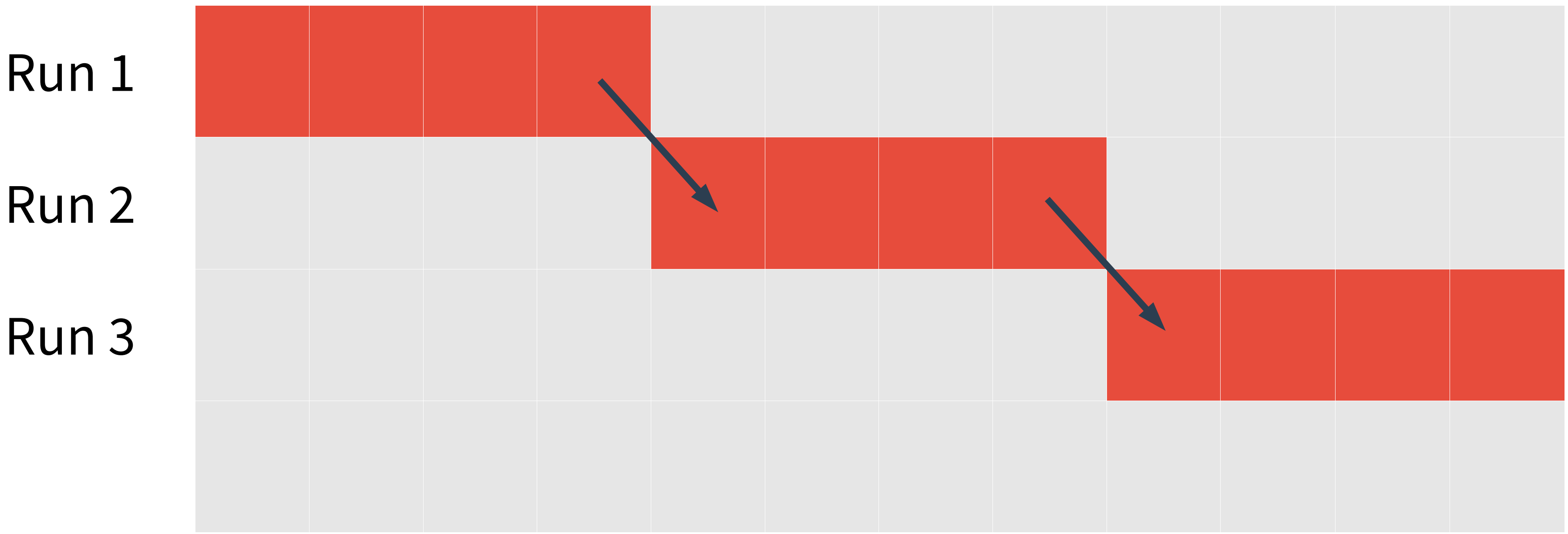
Communication Rounds (naive)



Communication Rounds (naive)

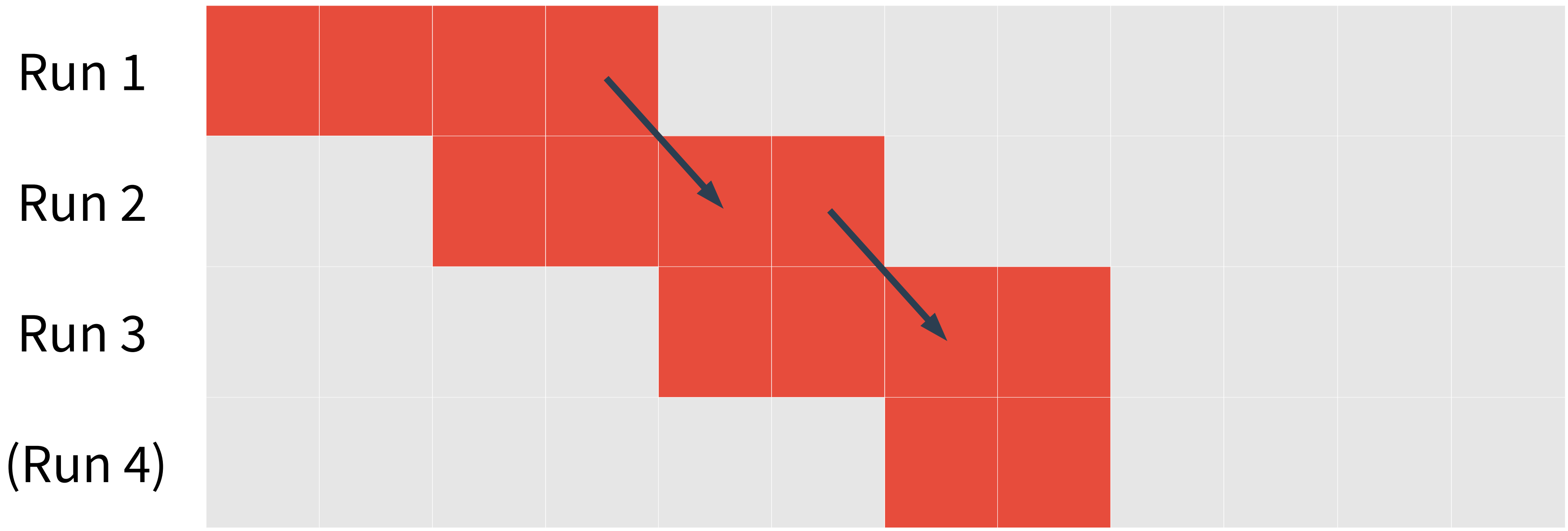


Communication Rounds (naive)



$4 + 4f$ rounds

Communication Rounds (DiceMix)



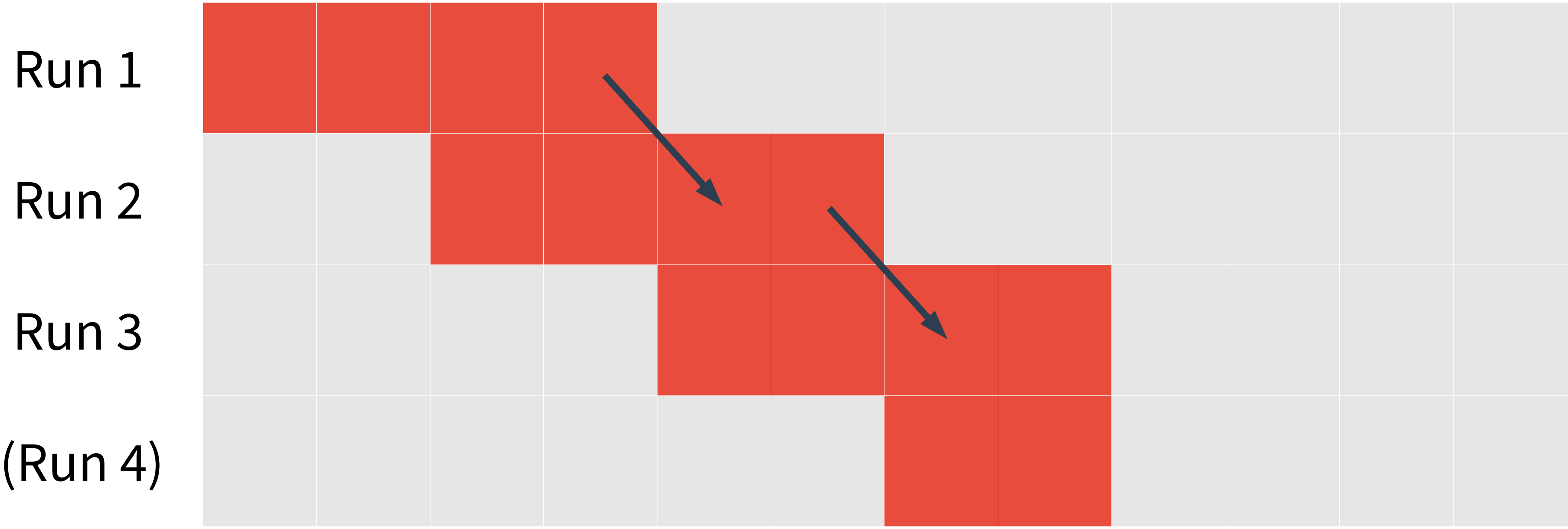
Run 1

Run 2

Run 3

(Run 4)

Communication Rounds (DiceMix)



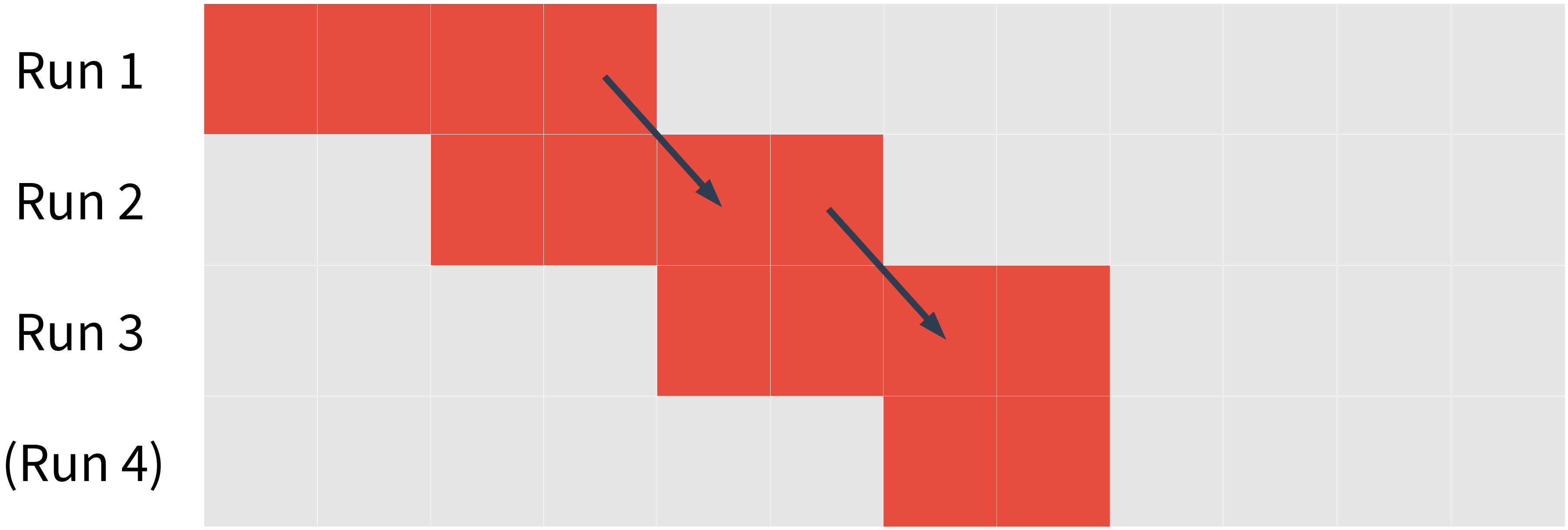
Run 1

Run 2

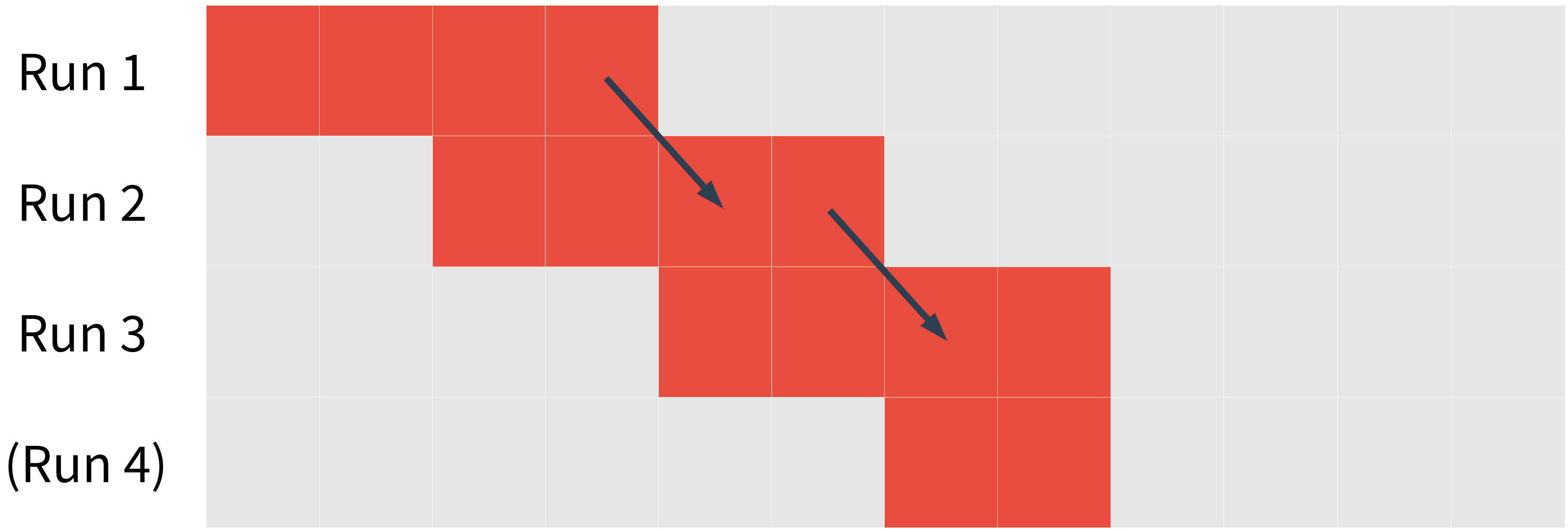
Run 3

(Run 4)

Communication Rounds (DiceMix)



Communication Rounds (DiceMix)



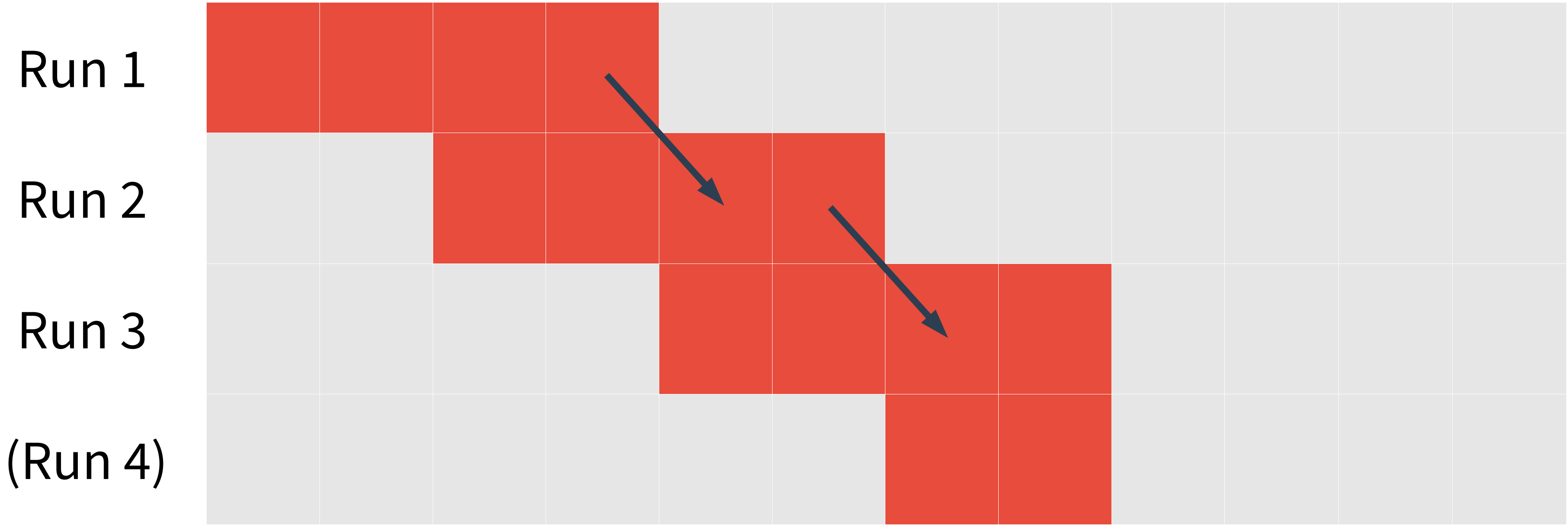
Run 1

Run 2

Run 3

(Run 4)

Communication Rounds (DiceMix)



$4 + 2f$ rounds

Communication Rounds (DiceMix)

Run 1	KE	DC	?	SK								
Run 2			KE	DC	? RV	🟥						
Run 3					🟥	🟥	🟥	🟥				
(Run 4)							🟥	🟥				

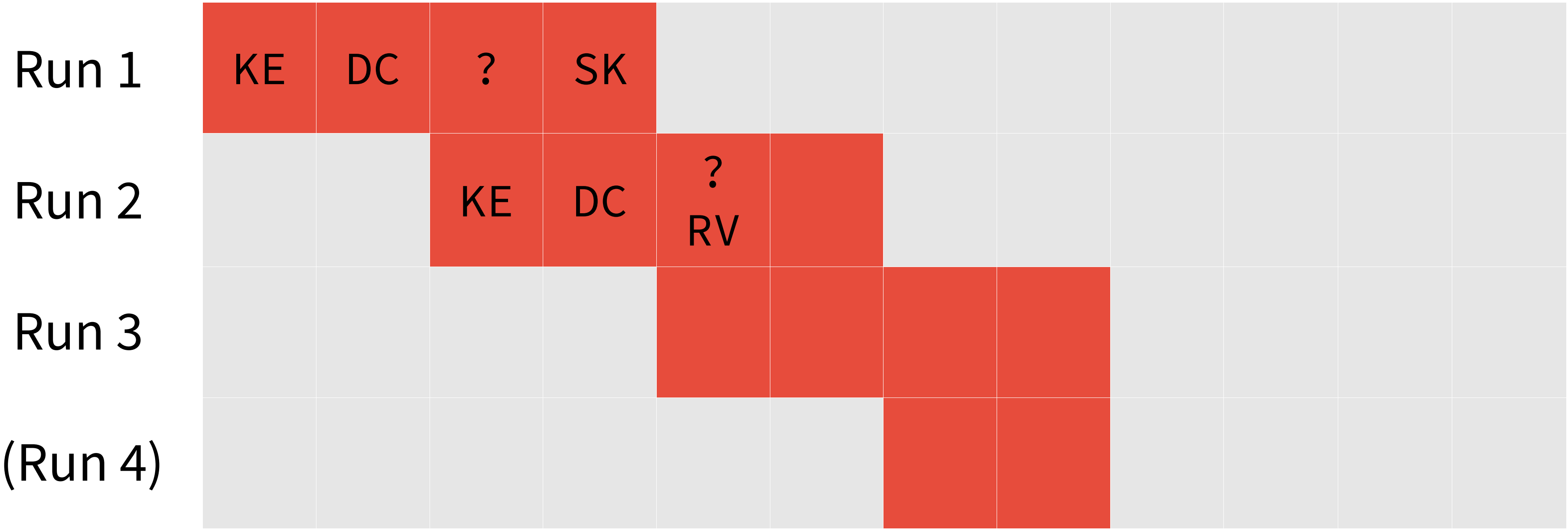
Communication Rounds (DiceMix)

Run 1	KE	DC	?	SK								
Run 2			KE	DC	? RV	🟥						
Run 3					🟥	🟥	🟥	🟥				
(Run 4)							🟥	🟥				

Communication Rounds (DiceMix)

Run 1	KE	DC	?	SK								
Run 2			KE	DC	? RV	🟥						
Run 3					🟥	🟥	🟥	🟥				
(Run 4)							🟥	🟥				

Communication Rounds (DiceMix)



Run 1	KE	DC	?	SK								
Run 2			KE	DC	? RV	🟥						
Run 3					🟥	🟥	🟥	🟥				
(Run 4)							🟥	🟥				

Communication Rounds (DiceMix)

Run 1	KE	DC	?	SK								
Run 2			KE	DC	? RV	🟥						
Run 3					🟥	🟥	🟥	🟥				
(Run 4)							🟥	🟥				

$$4 + 2f \ \text{ rounds}$$

DC-nets in Practice

DC-nets in Practice

- Key exchange to establish shared keys

DC-nets in Practice

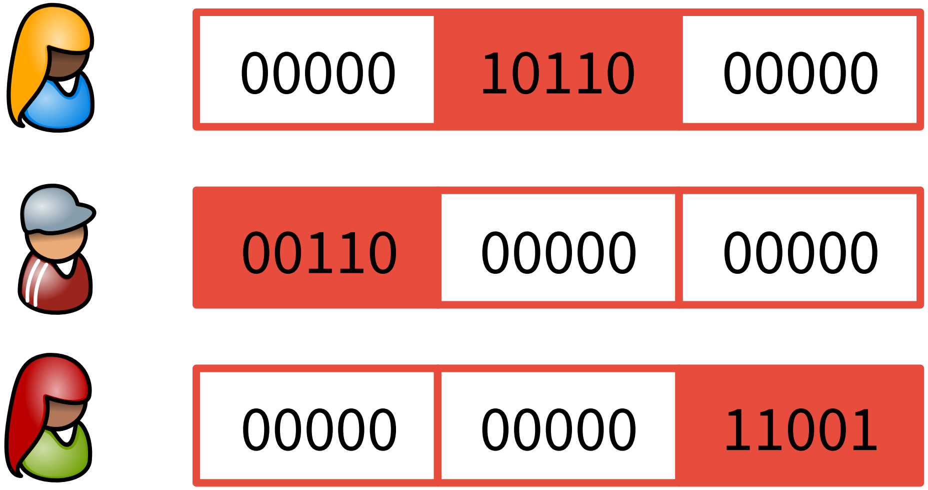
- Key exchange to establish shared keys
- Send bitstrings instead of single bits

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

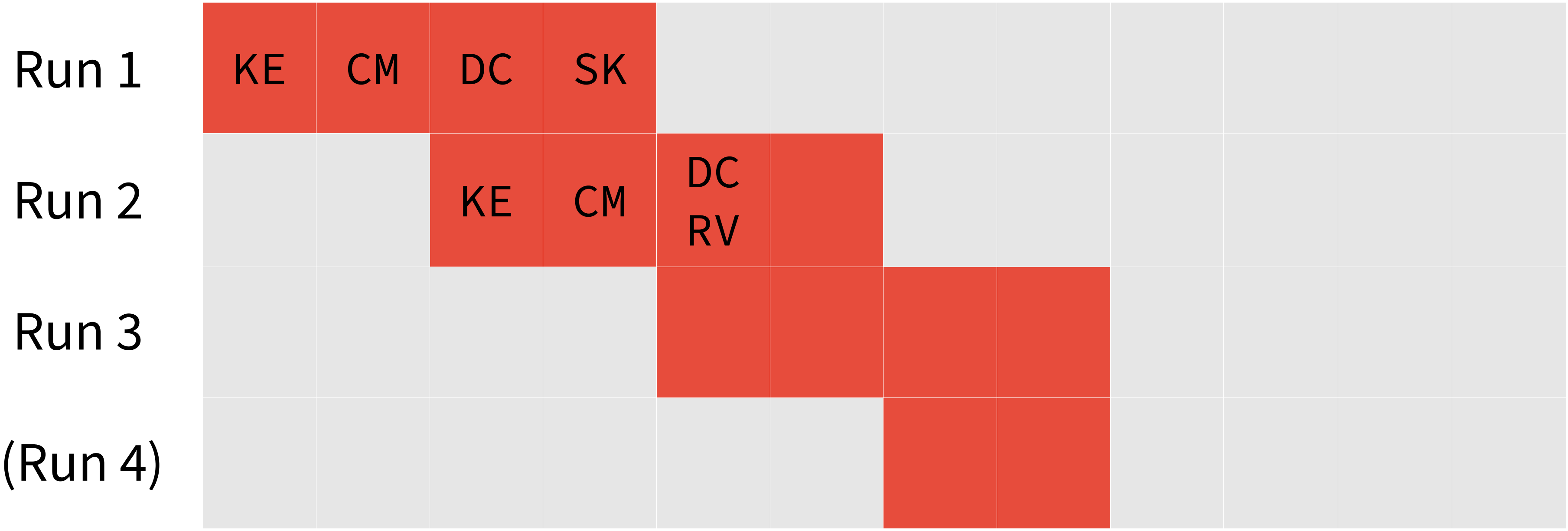


00000	10110	00000
00110	00000	00000
00000	00000	11001

Communication Rounds (DiceMix)

Run 1	KE	CM	DC	SK								
Run 2			KE	CM	DC RV	🟥						
Run 3					🟥	🟥	🟥	🟥				
(Run 4)							🟥	🟥				

Communication Rounds (DiceMix)



Run 1	KE	CM	DC	SK								
Run 2			KE	CM	DC RV	🟥						
Run 3					🟥	🟥	🟥	🟥				
(Run 4)							🟥	🟥				

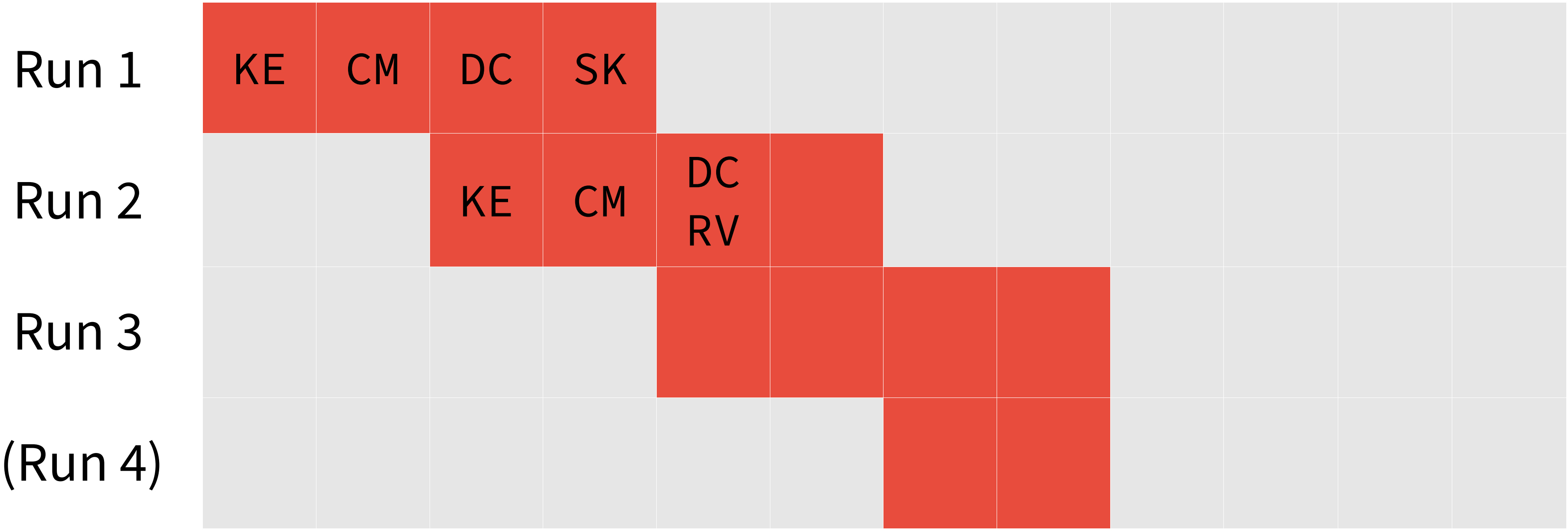
Communication Rounds (DiceMix)

Run 1	KE	CM	DC	SK								
Run 2			KE	CM	DC RV	🟥						
Run 3					🟥	🟥	🟥	🟥				
(Run 4)							🟥	🟥				

Communication Rounds (DiceMix)

Run 1	KE	CM	DC	SK								
Run 2			KE	CM	DC RV	🟥						
Run 3					🟥	🟥	🟥	🟥				
(Run 4)							🟥	🟥				

Communication Rounds (DiceMix)



Run 1	KE	CM	DC	SK								
Run 2			KE	CM	DC RV	🟥						
Run 3					🟥	🟥	🟥	🟥				
(Run 4)							🟥	🟥				

$4 + 2f$ rounds