

P2P Mixing and Unlinkable Bitcoin Transactions

Tim Ruffing

@real_or_random

Pedro Moreno-Sanchez

@pedrorechez

Aniket Kate

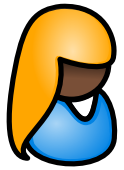
@aniketpkate



UNIVERSITÄT
DES
SAARLANDES

PURDUE
UNIVERSITY

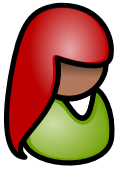
P2P Mixing



— A



— B

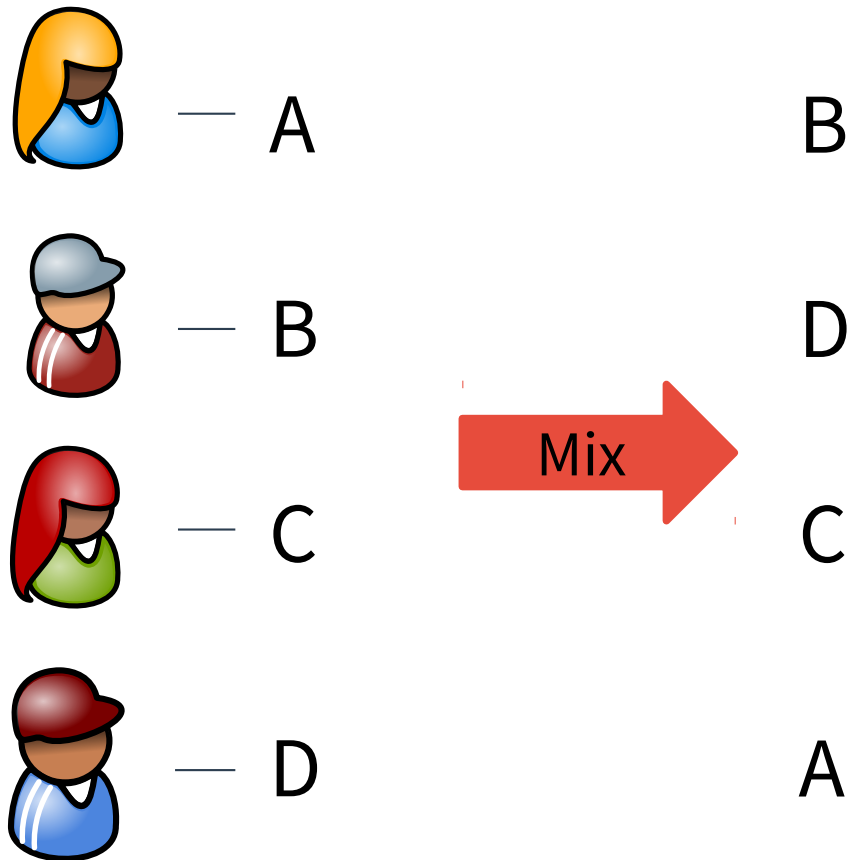


— C

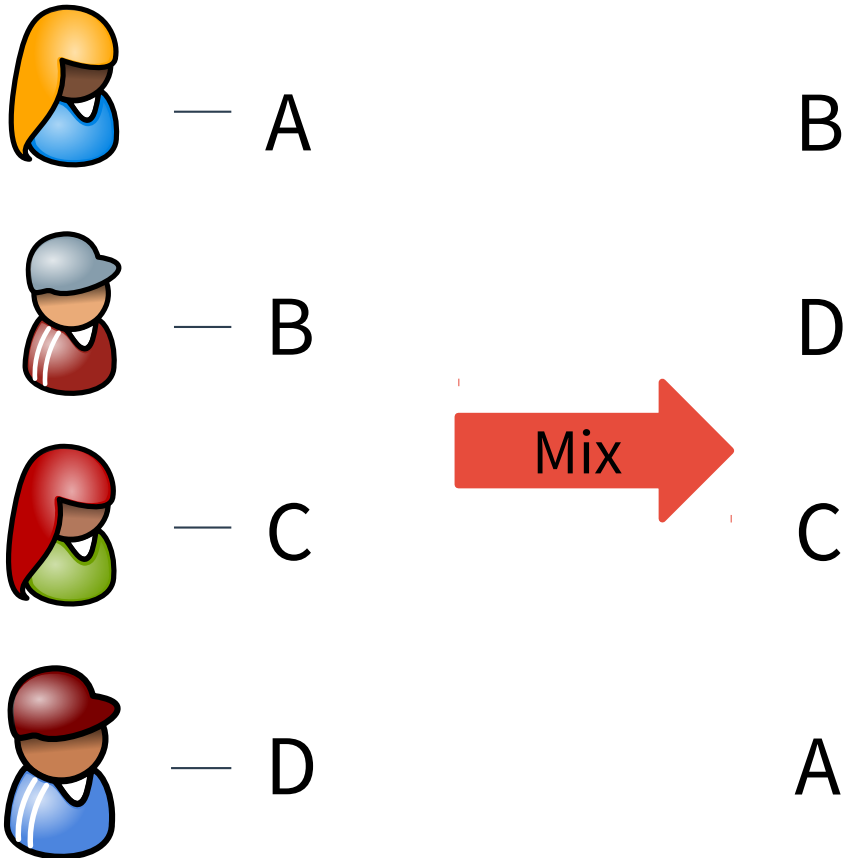


— D

P2P Mixing



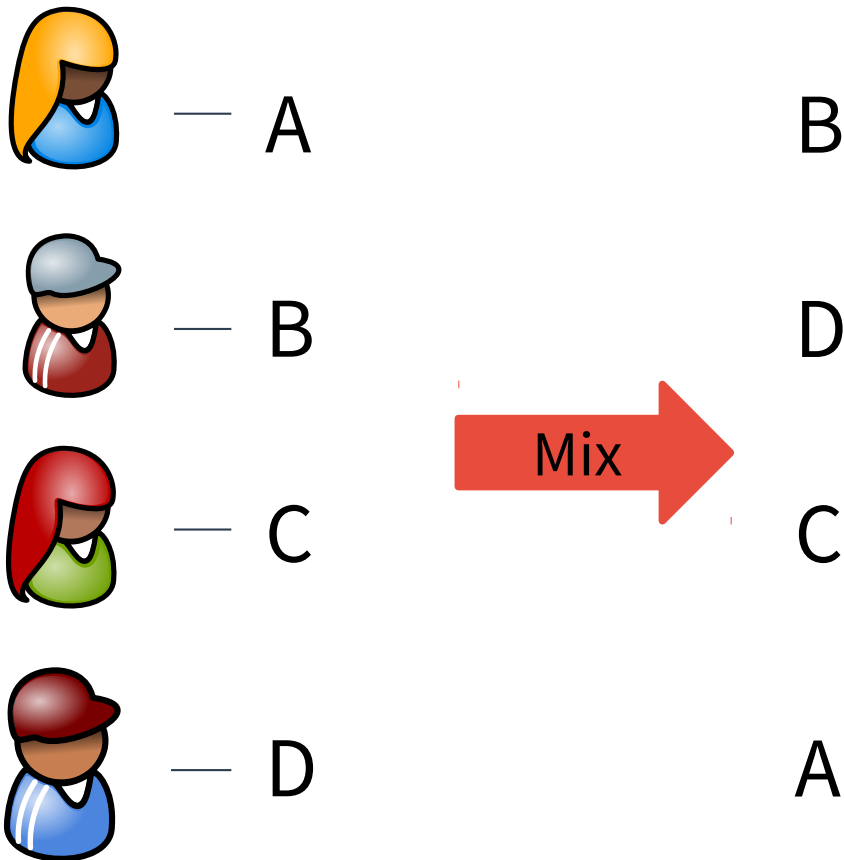
P2P Mixing



Confirmation

- Peers agree on the output and confirm it

P2P Mixing



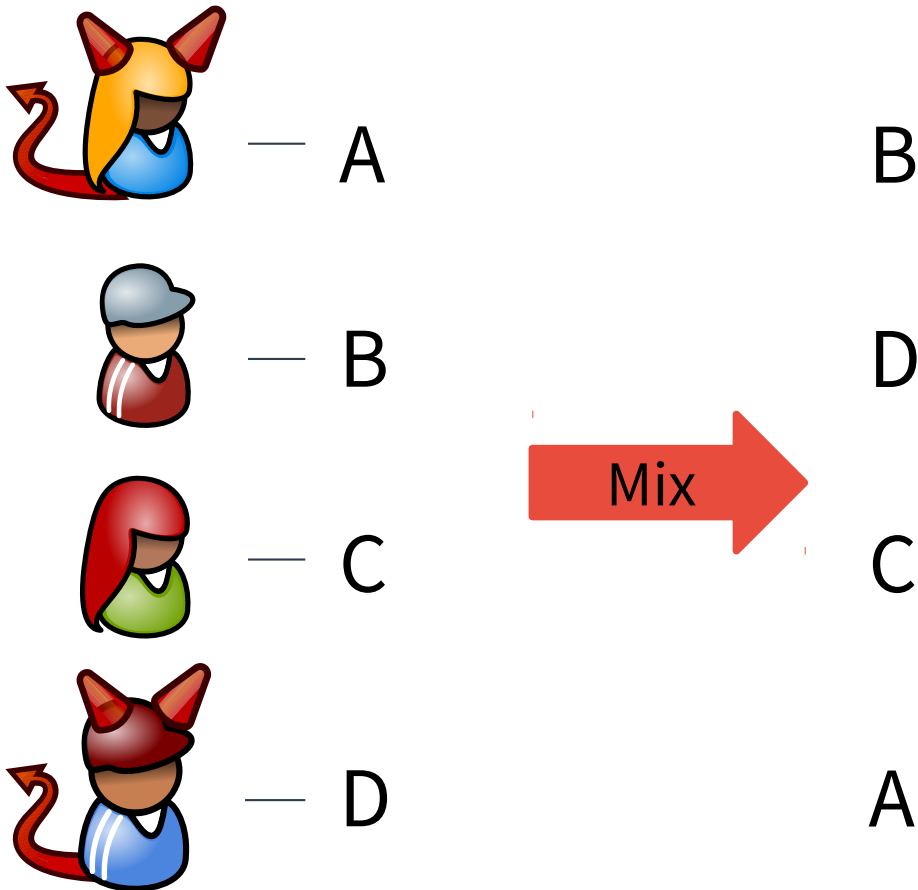
Confirmation

- Peers agree on the output and confirm it

P2P Trust model

- No mutual trust, no third-party routers

P2P Mixing



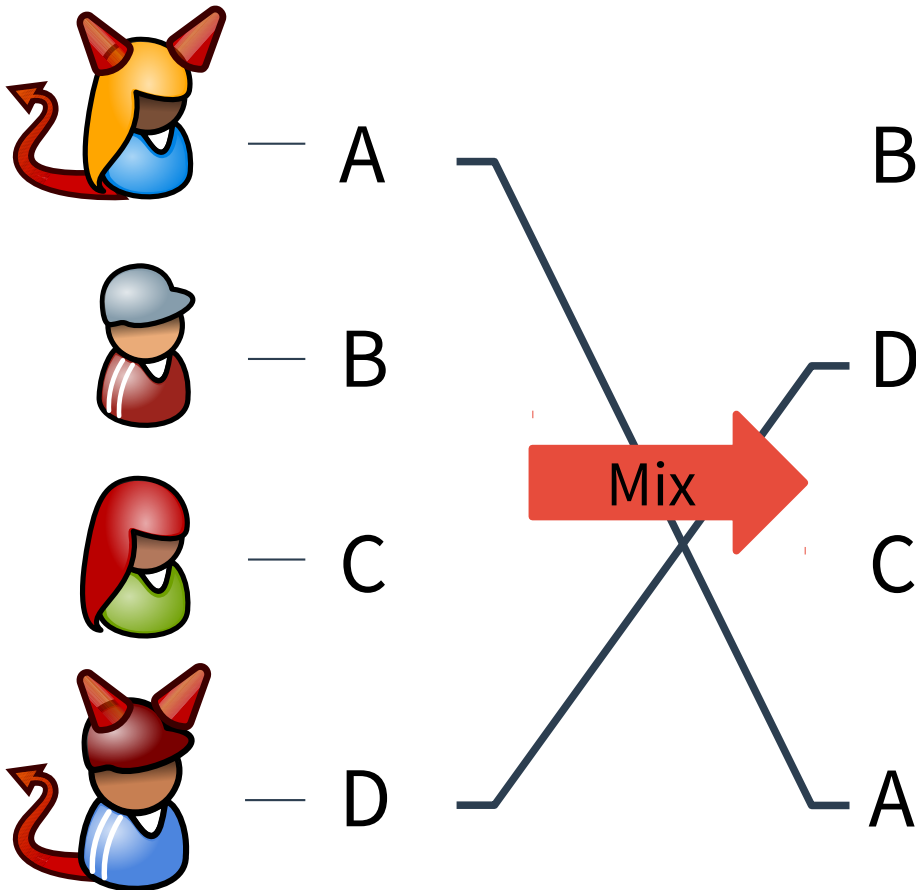
Confirmation

- Peers agree on the output and confirm it

P2P Trust model

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

P2P Mixing



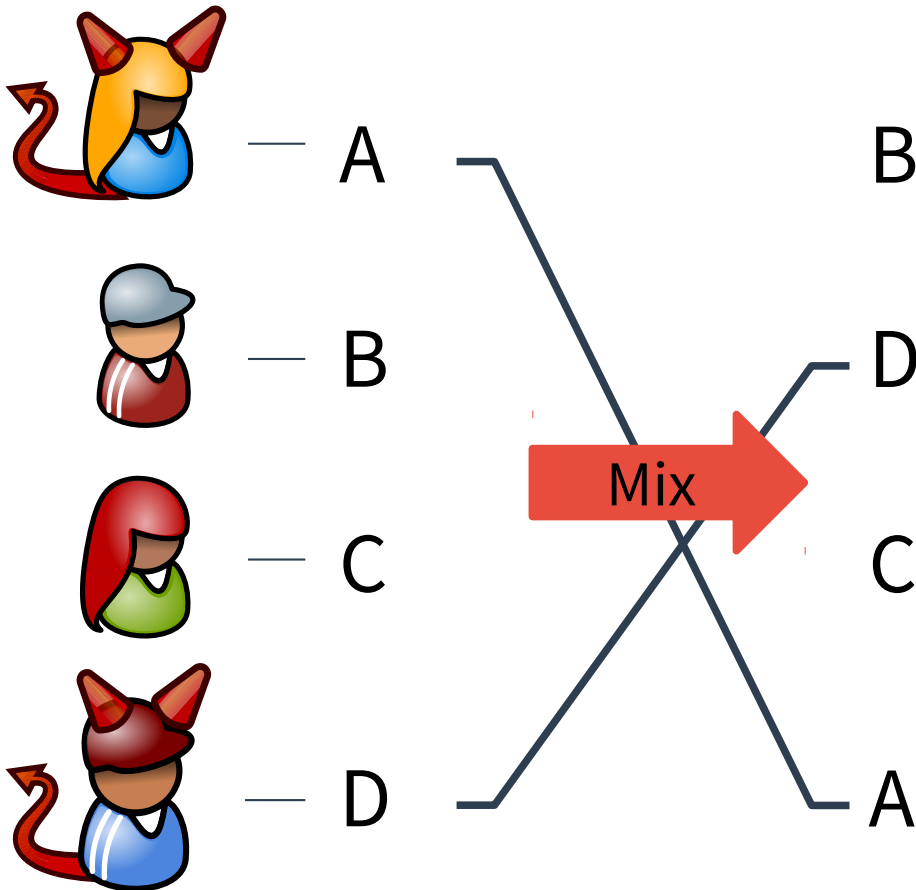
Confirmation

- Peers agree on the output and confirm it

P2P Trust model

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

P2P Mixing



Confirmation

- Peers agree on the output and confirm it

P2P Trust model

- No mutual trust, no third-party routers
- Anonymity set is the set of honest users
- Protocol must terminate in the presence of $f < n - 1$ malicious users

State of the Art (I)

State of the Art (I)

Traditional mixnet run by all peers

State of the Art (I)

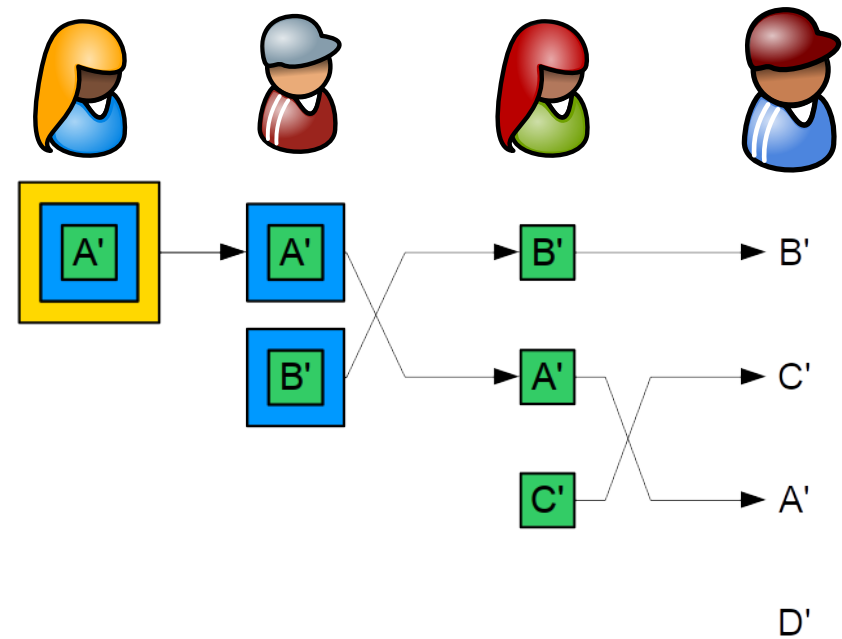
Traditional mixnet run by all peers

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

State of the Art (I)

Traditional mixnet run by all peers

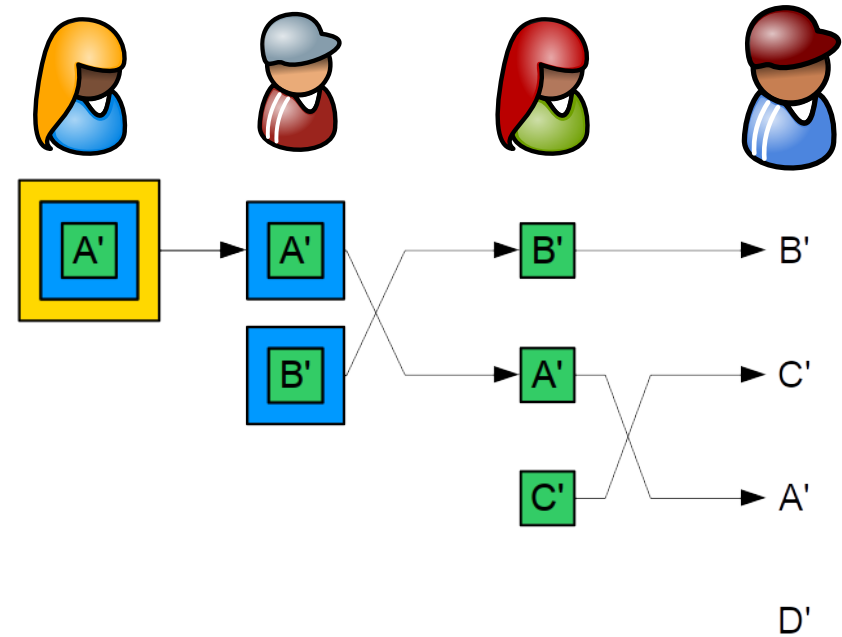
- Dissent (shuffle protocol) [CCS 2010],
CoinShuffle [ESORICS 2014]



State of the Art (I)

Traditional mixnet run by all peers

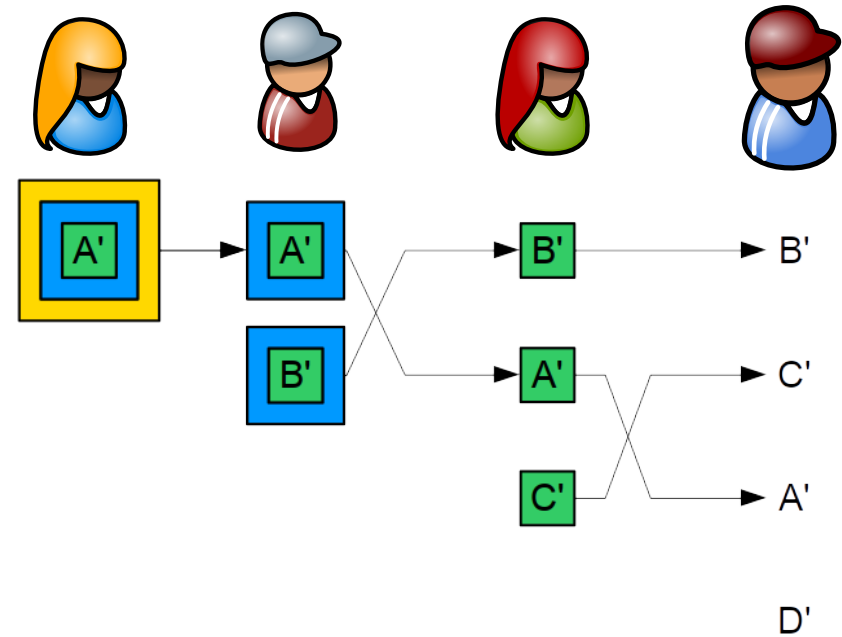
- Dissent (shuffle protocol) [CCS 2010], CoinShuffle [ESORICS 2014]
- $O(n)$ rounds in optimistic case



State of the Art (I)

Traditional mixnet run by all peers

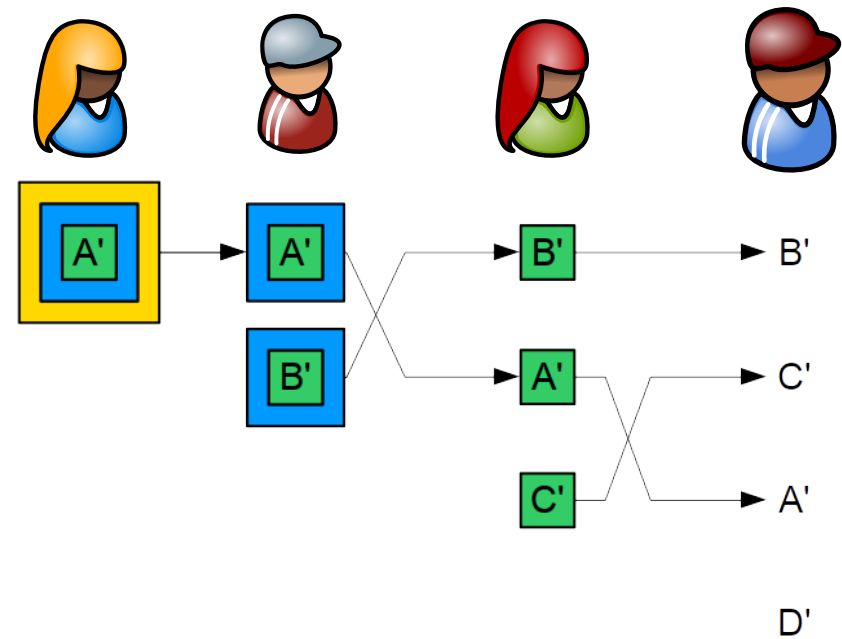
- Dissent (shuffle protocol) [CCS 2010], CoinShuffle [ESORICS 2014]
- $O(n)$ rounds in optimistic case
- $O(nf)$ rounds for f malicious peers



State of the Art (I)

Traditional mixnet run by all peers

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



Traditional mixnet solution does not scale!

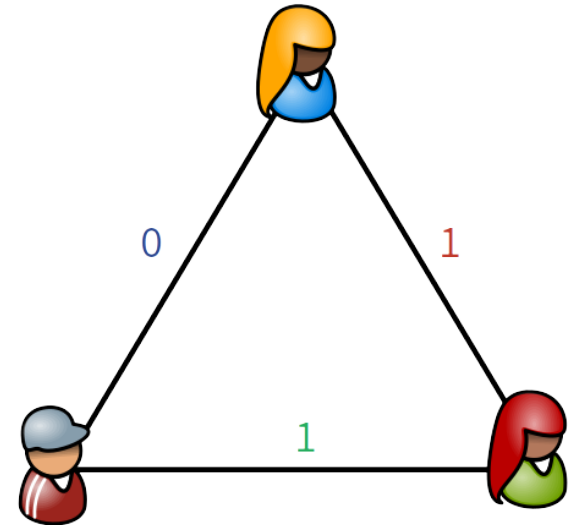
State of the Art (II)

State of the Art (II)

Dining cryptographers' networks (DC-nets)

State of the Art (II)

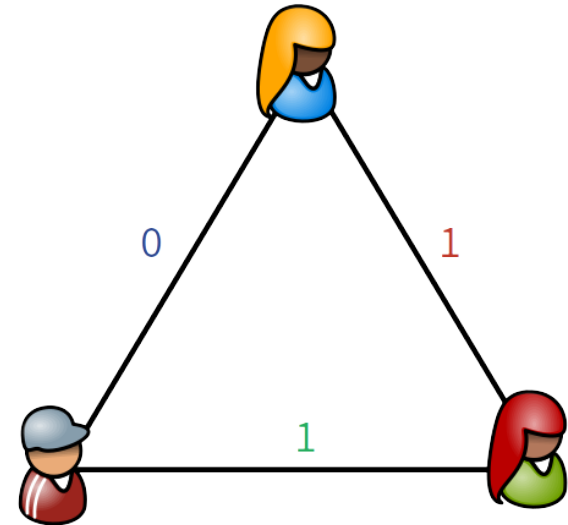
Dining cryptographers' networks (DC-nets)



State of the Art (II)

Dining cryptographers' networks (DC-nets)

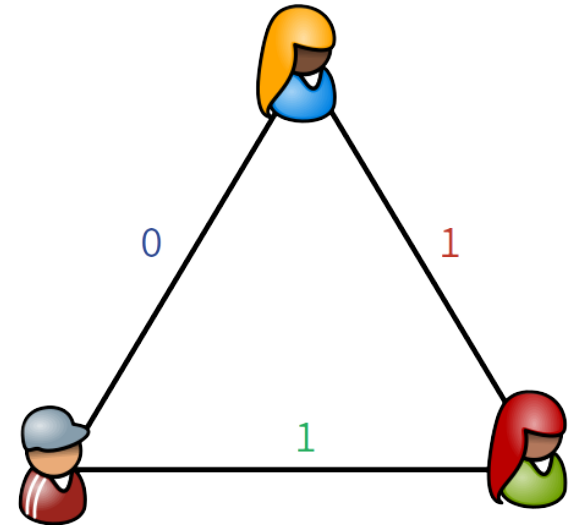
- Hope for $O(1)$ rounds in the optimistic case



State of the Art (II)

Dining cryptographers' networks (DC-nets)

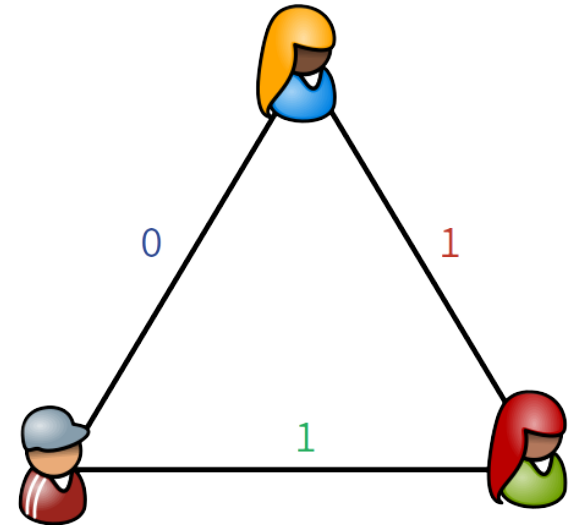
- Hope for $O(1)$ rounds in the optimistic case
- Easy to disrupt



State of the Art (II)

Dining cryptographers' networks (DC-nets)

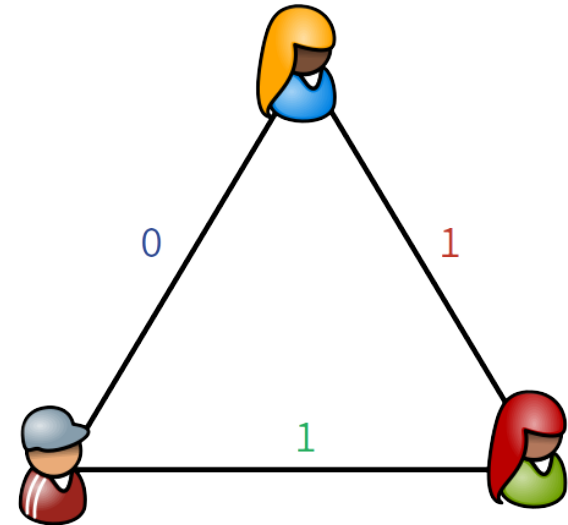
- Hope for $O(1)$ rounds in the optimistic case
- Easy to disrupt
- All approaches to solve disruption problem suffer from drawbacks



State of the Art (II)

Dining cryptographers' networks (DC-nets)

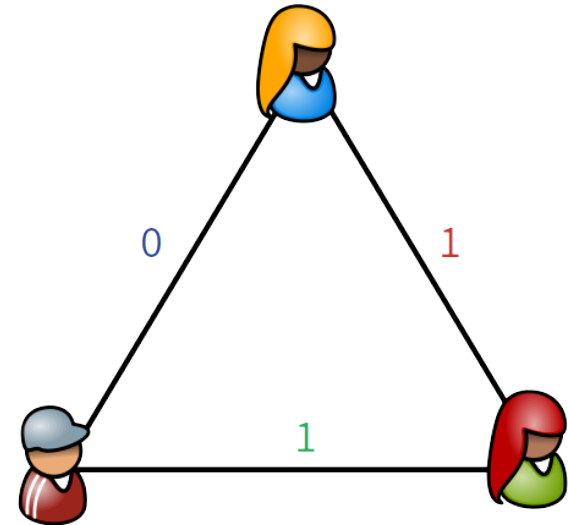
- Hope for $O(1)$ rounds in the optimistic case
- Easy to disrupt
- All approaches to solve disruption problem suffer from drawbacks
- Golle and Juels [EUROCRYPT 2004]:
Honest majority



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!

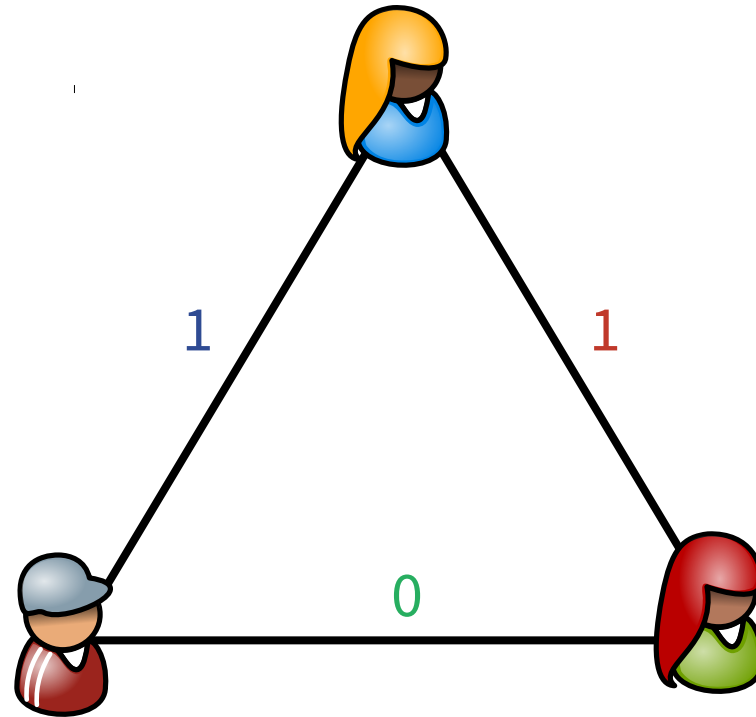
DiceMix

A Practical P2P Mixing Protocol based on DC-nets

DC-net [Chaum 1988]

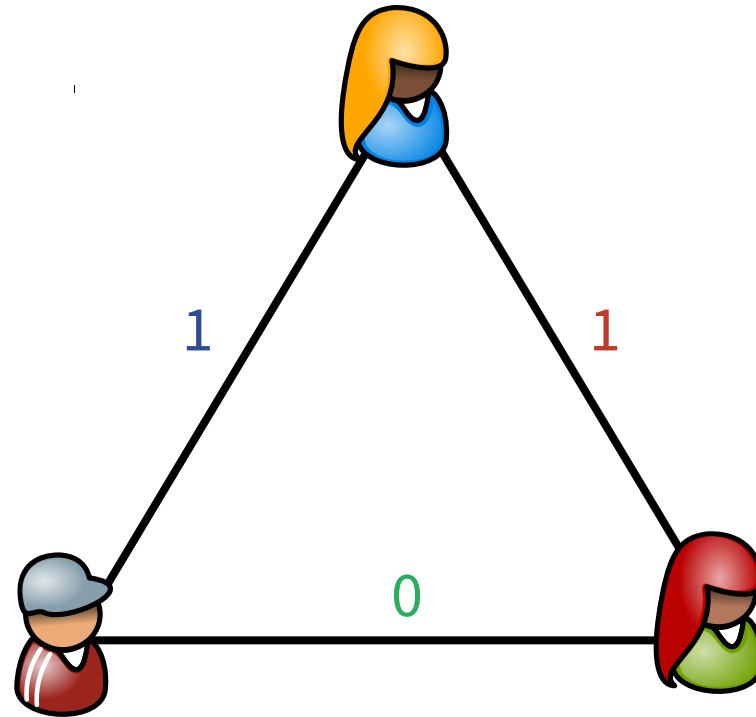


DC-net [Chaum 1988]

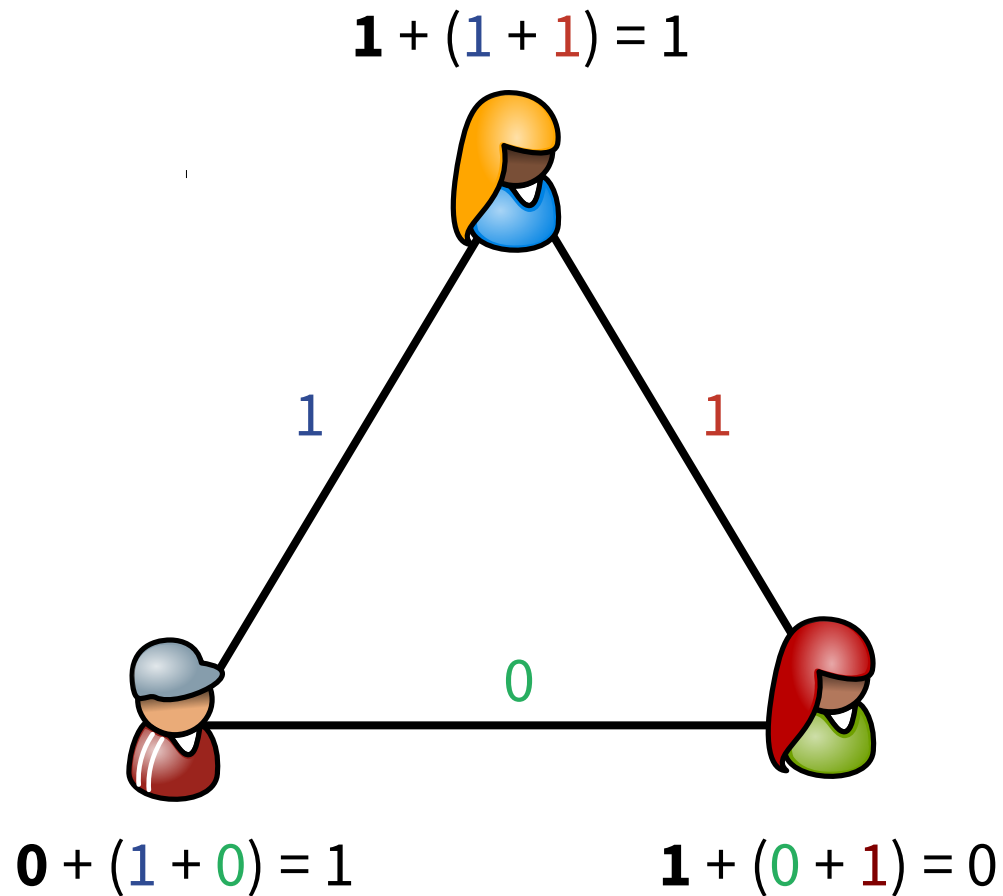


DC-net [Chaum 1988]

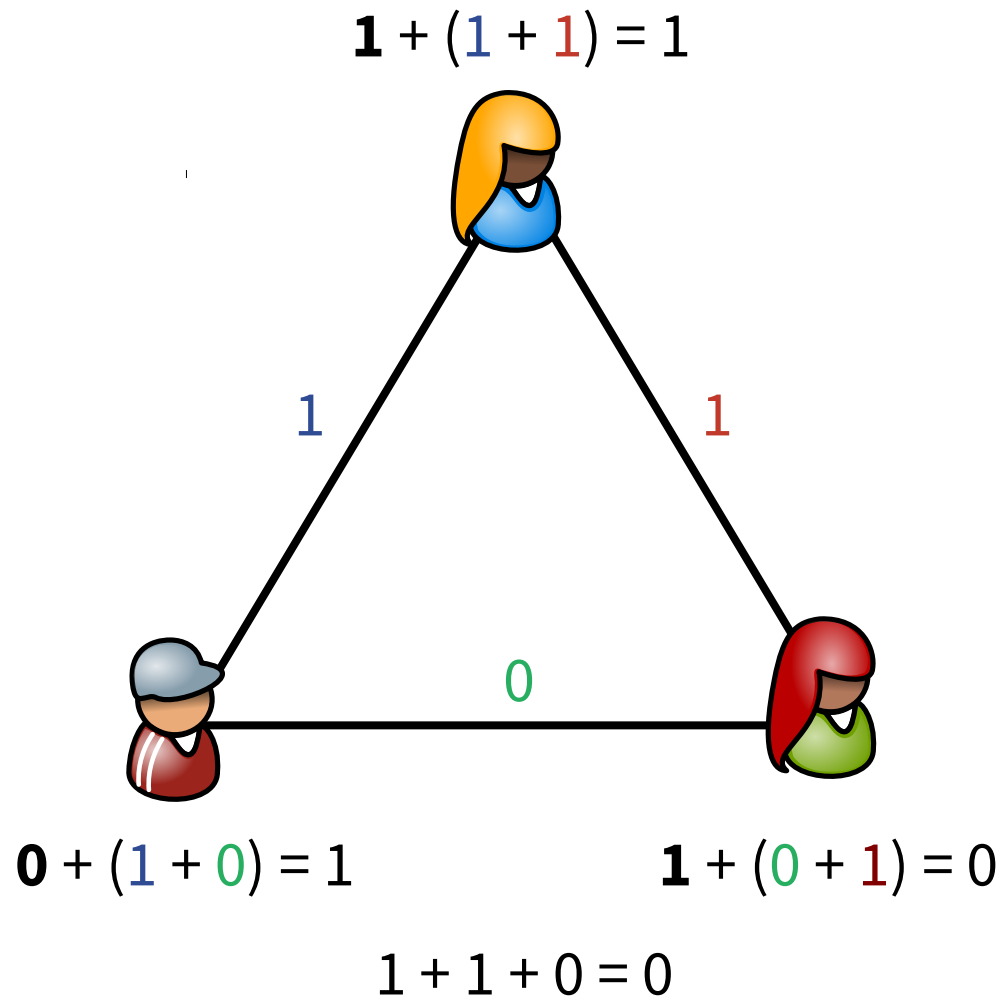
$$1 + (1 + 1) = 1$$



DC-net [Chaum 1988]



DC-net [Chaum 1988]



Sending Several Messages [Bos, Boer 1989]



User 1:

m_1



User 2:

m_2



User 3:

m_3

⋮



User n :

m_n

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

Sending Several Messages [Bos, Boer 1989]



User 1:

m_1

m_1^2

m_1^3

...

m_1^n



User 2:

m_2

m_2^2

m_2^3

...

m_2^n



User 3:

m_3

m_3^2

m_3^3

...

m_3^n

⋮

⋮

⋮

⋮

⋮



User n :

m_n

m_n^2

m_n^3

...

m_n^n

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

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

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

...

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

Sending Several Messages [Bos, Boer 1989]



User 1:

m_1

m_1^2

m_1^3

...

m_1^n



User 2:

m_2

m_2^2

m_2^3

...

m_2^n



User 3:

m_3

m_3^2

m_3^3

...

m_3^n

⋮

⋮

⋮

⋮

⋮



User n :

m_n

m_n^2

m_n^3

...

m_n^n

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

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





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

...

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

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

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:

User 2:



User 3:



User n :

m_1	m_1^2	m_1^3	...	m_1^n
m_1	m_2^2	m_2^3	...	m_2^n
m_3	m_3^2	m_3^3	...	m_3^n
\vdots	\vdots	\vdots	\ddots	\vdots
m_n	m_n^2	m_n^3	...	m_n^n
			...	

Disruption



User 1:

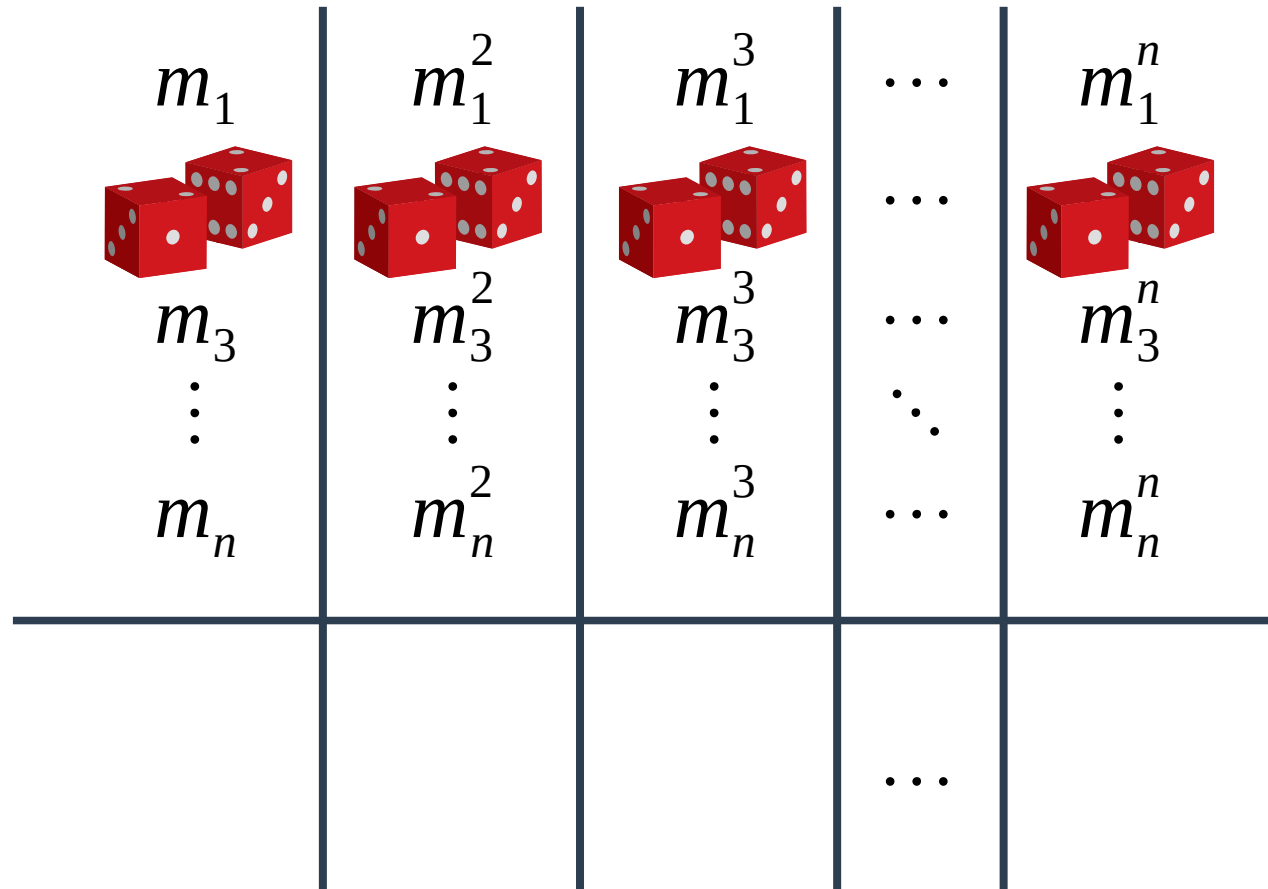
User 2:



User 3:



User n :



Disruption



User 1:

m_1



User 2:



...



User 3:

m_3

⋮

m_3^2

⋮

m_3^3

⋮

...

m_3^n

⋮

User n :

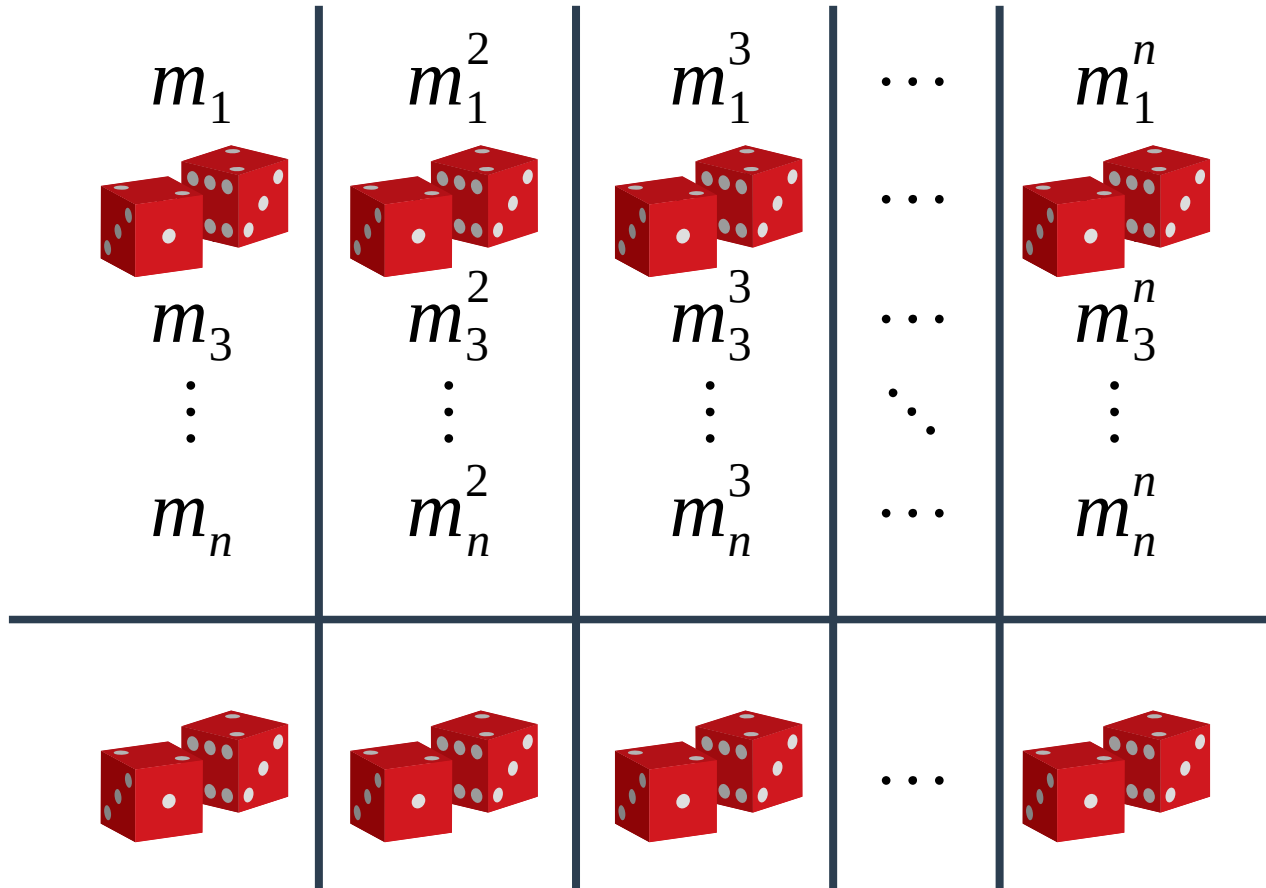
m_n

m_n^2

m_n^3

...

m_n^n



Malicious user stays anonymous!

Handling Disruptions

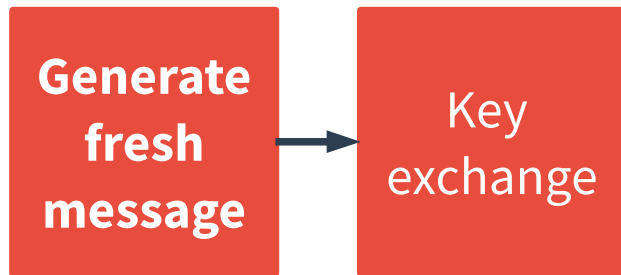
**IN CASE OF
DISRUPTION
BREAK ANONYMITY**



Flowchart of DiceMix

**Generate
fresh
message**

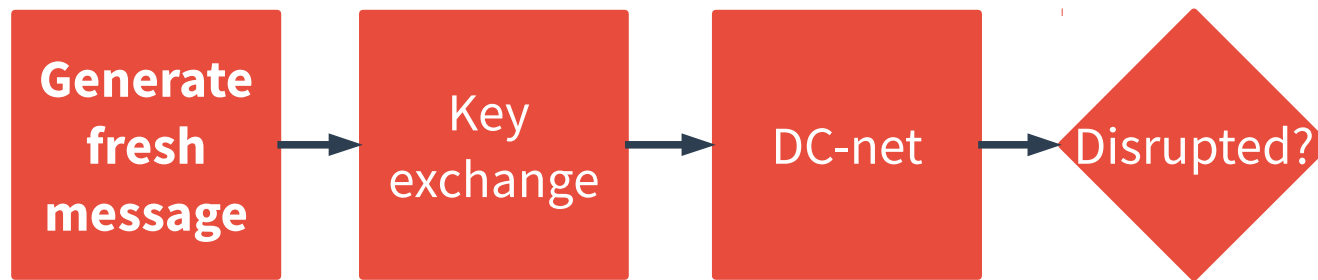
Flowchart of DiceMix



Flowchart of DiceMix



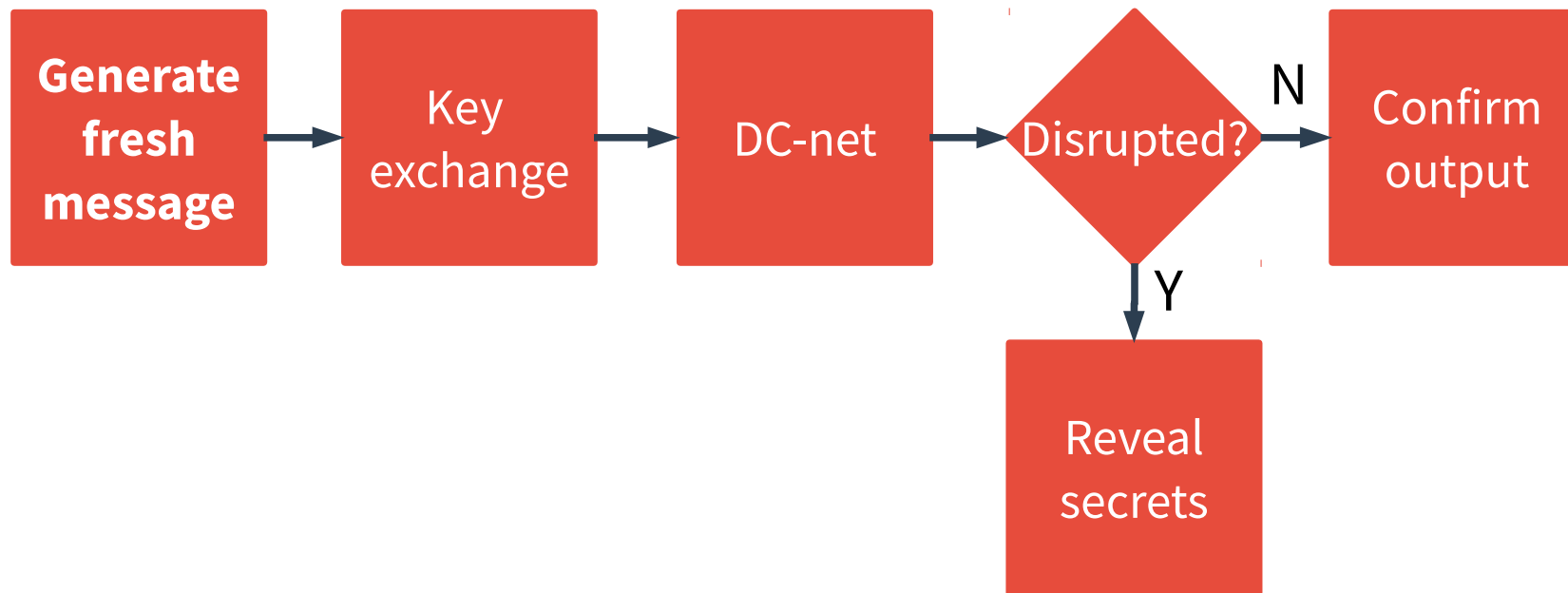
Flowchart of DiceMix



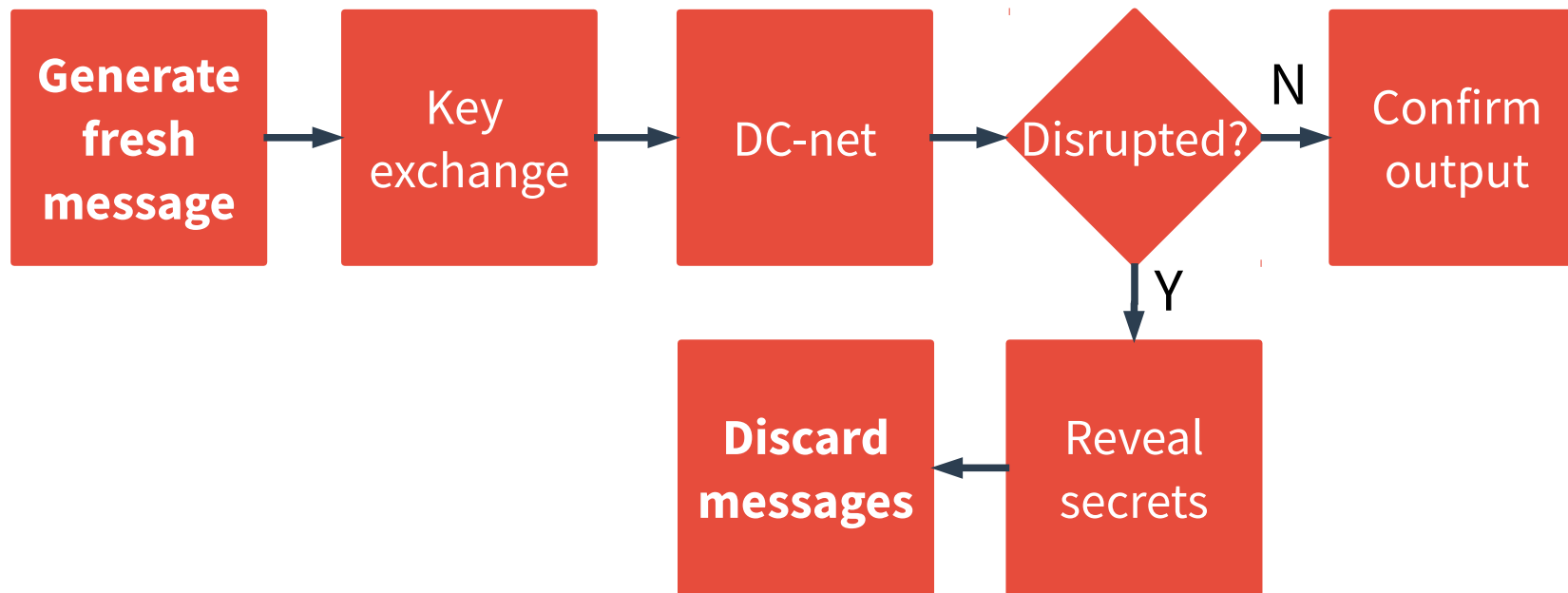
Flowchart of DiceMix



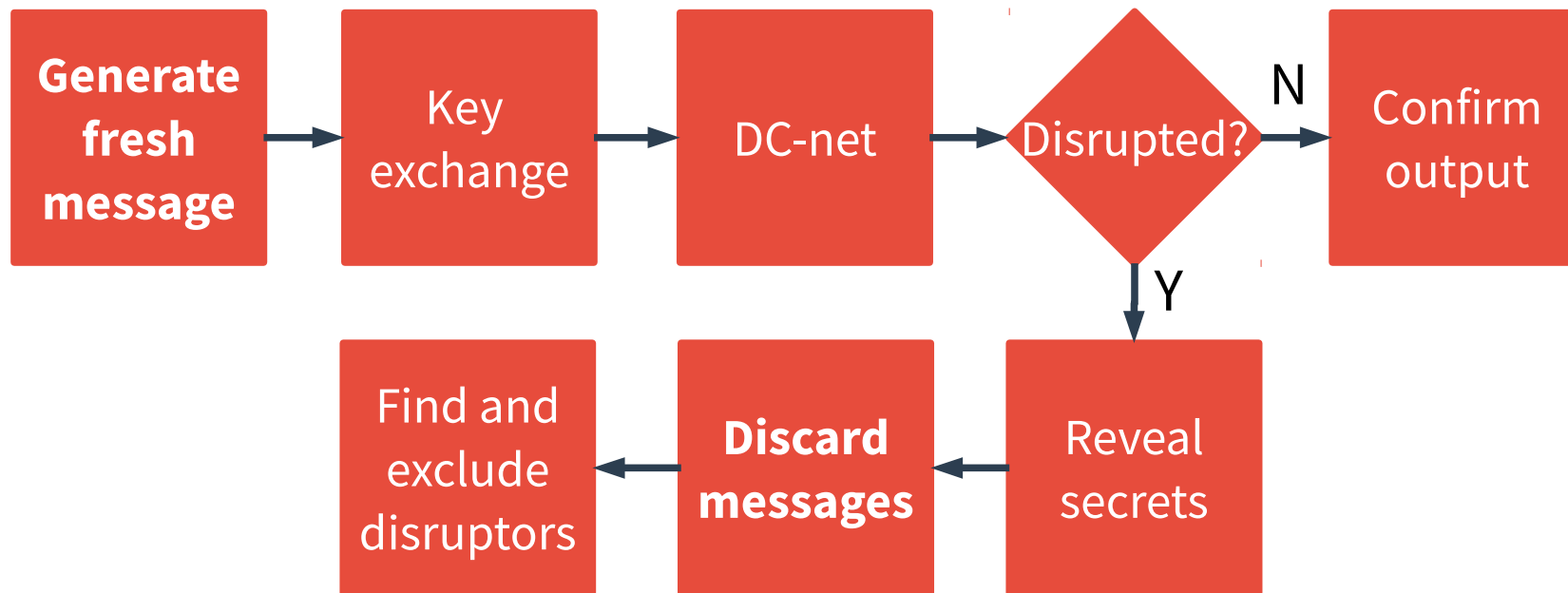
Flowchart of DiceMix



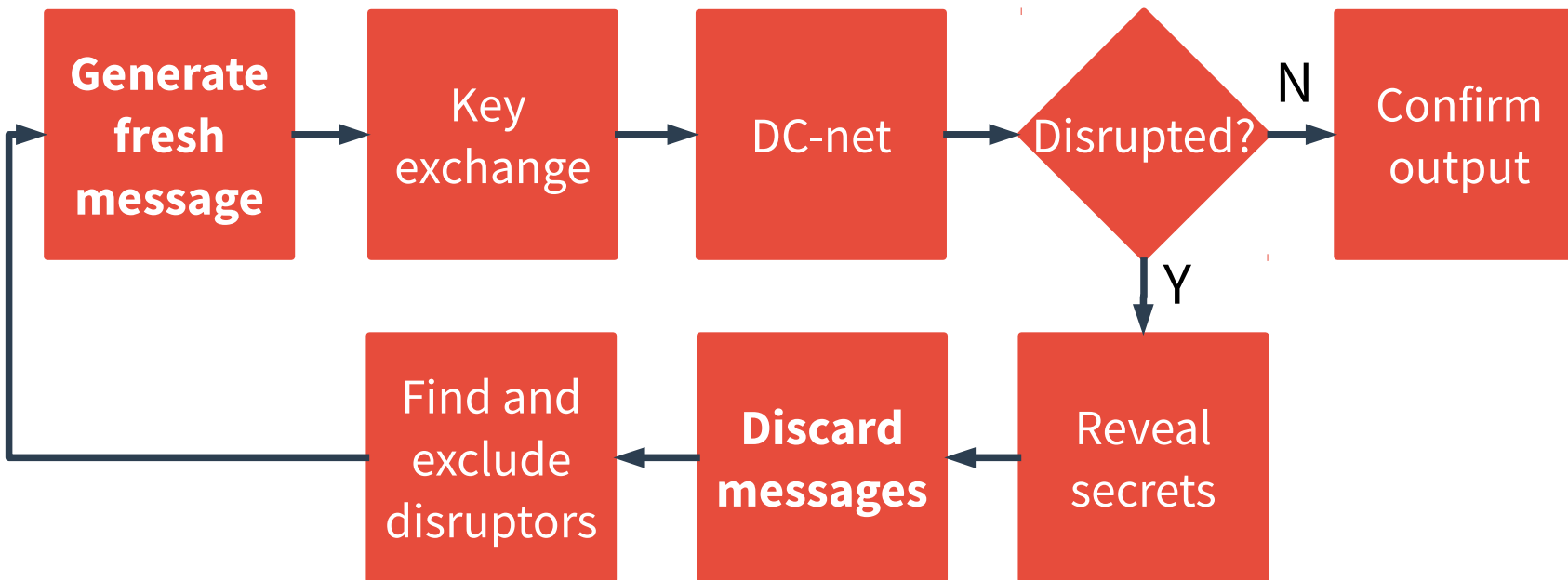
Flowchart of DiceMix



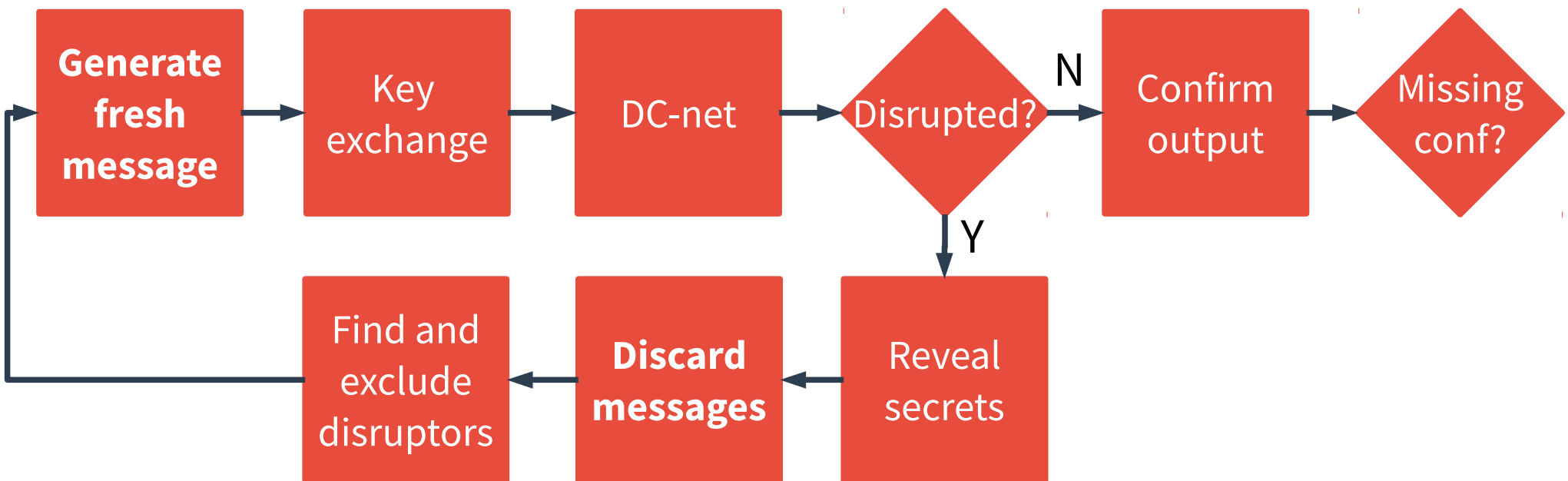
Flowchart of DiceMix



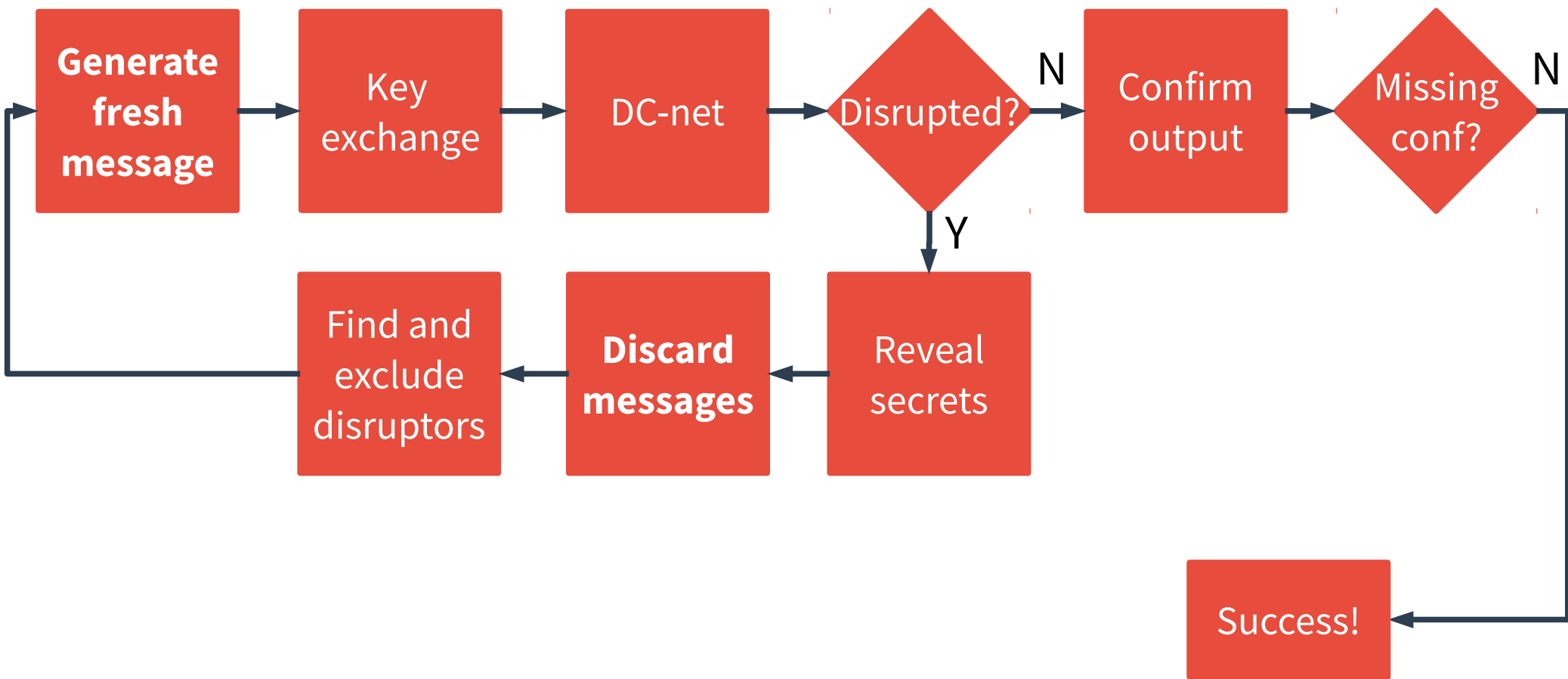
Flowchart of DiceMix



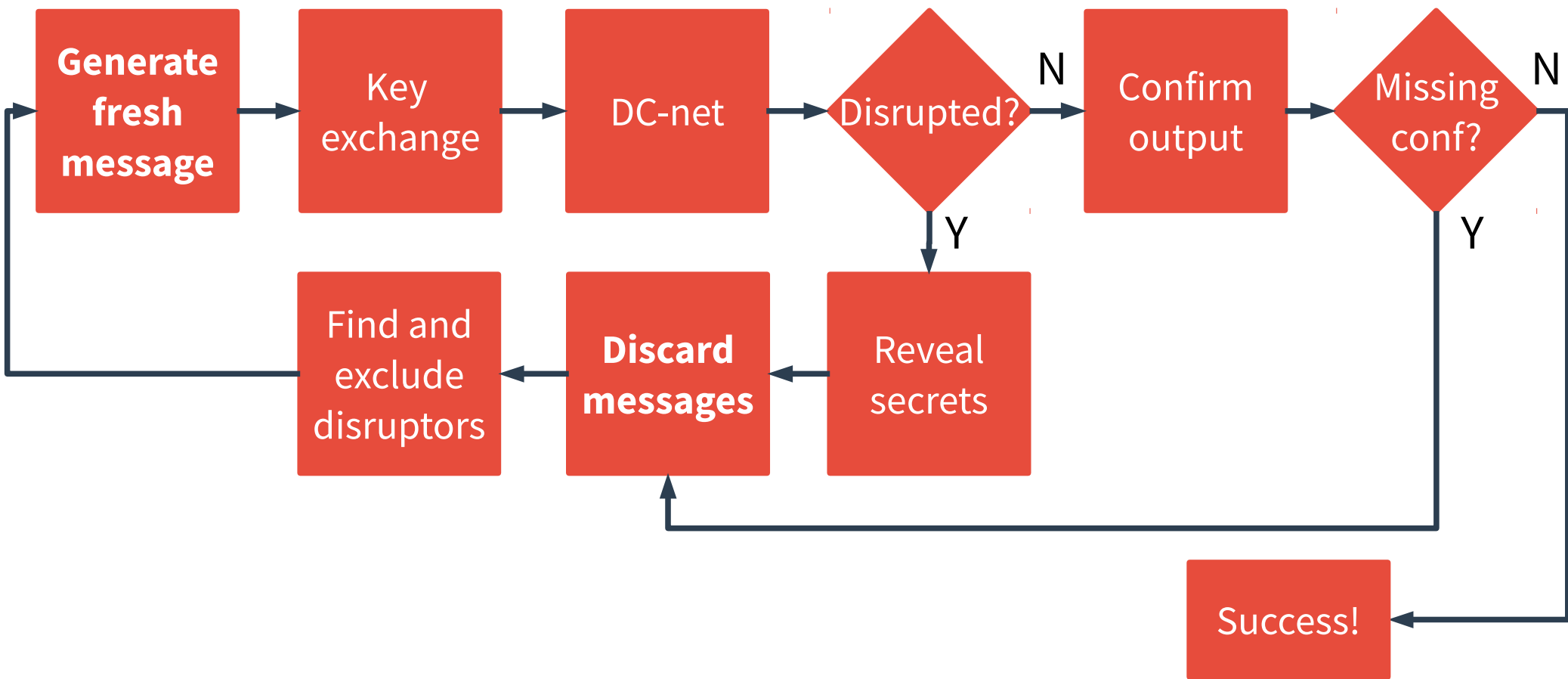
Flowchart of DiceMix



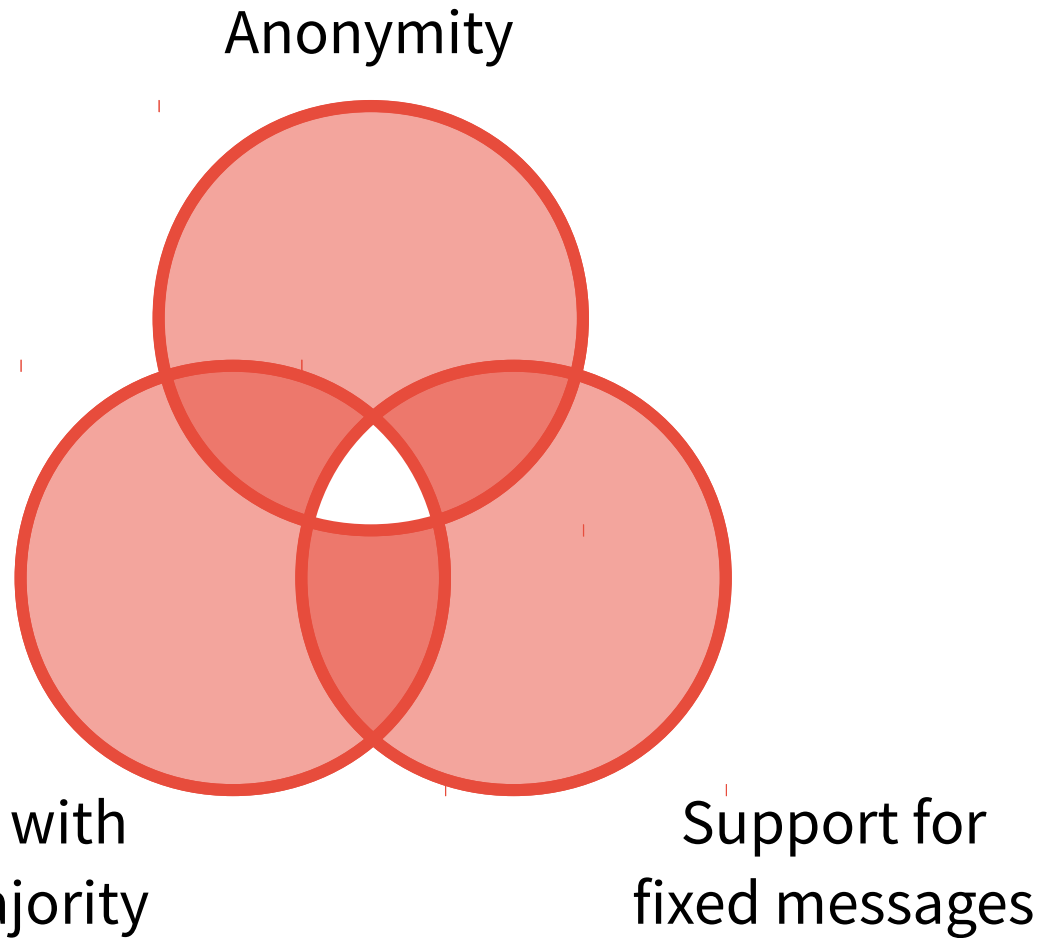
Flowchart of DiceMix



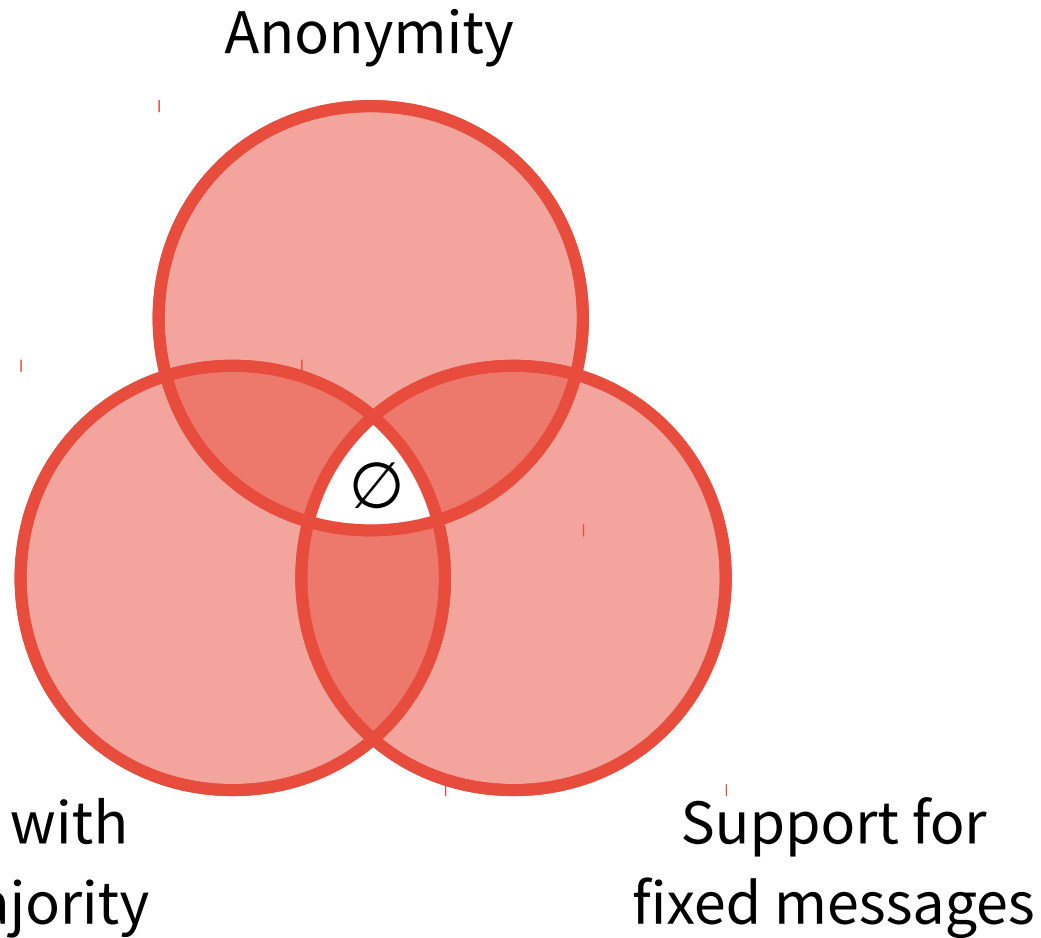
Flowchart of DiceMix



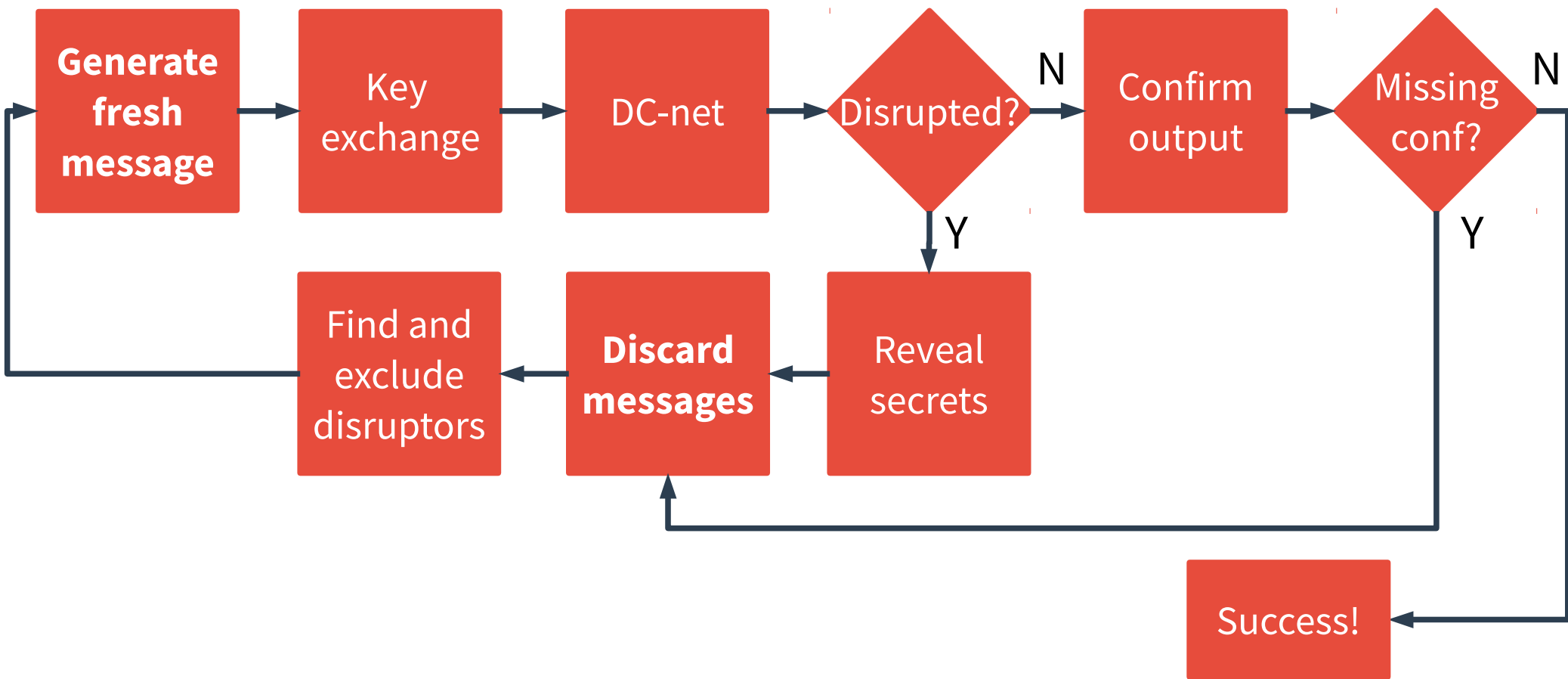
Freshness Is Necessary



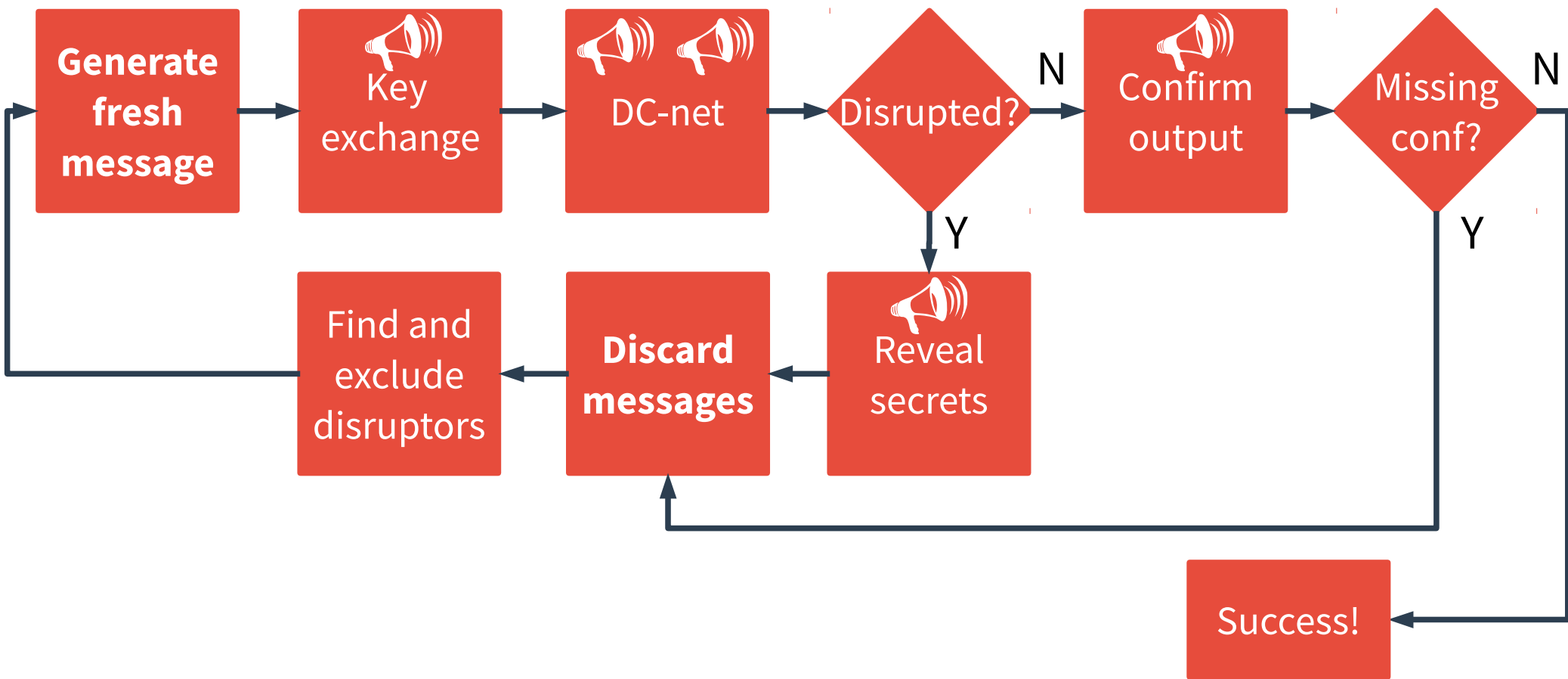
Freshness Is Necessary



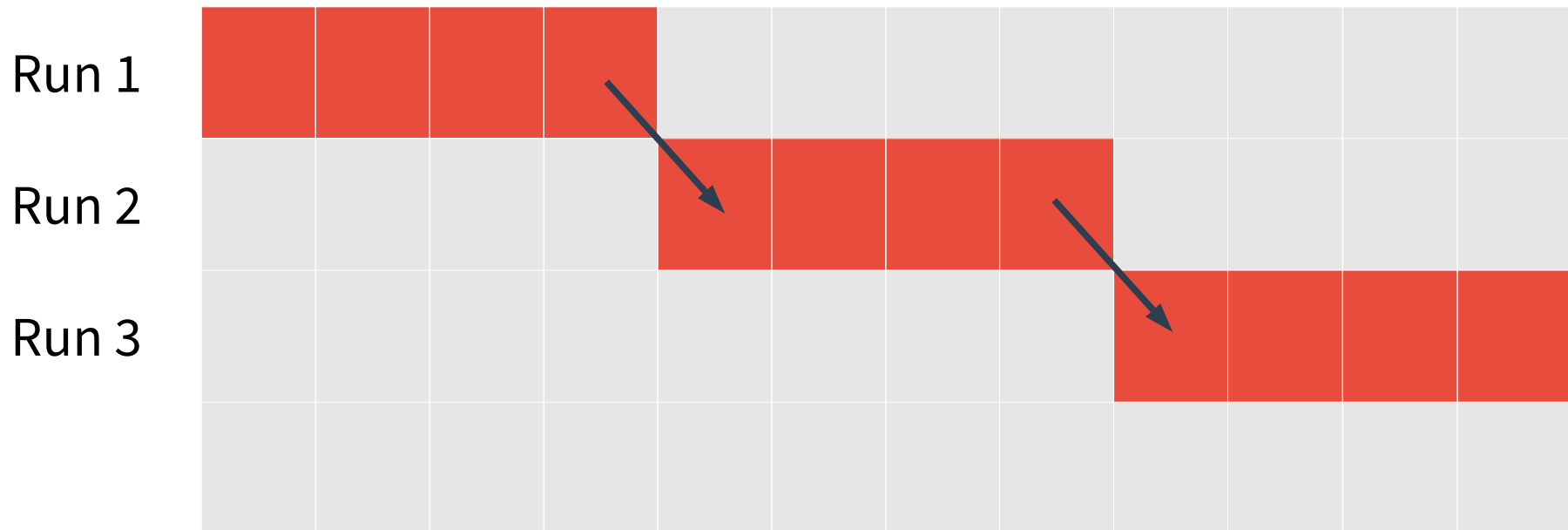
Flowchart of DiceMix



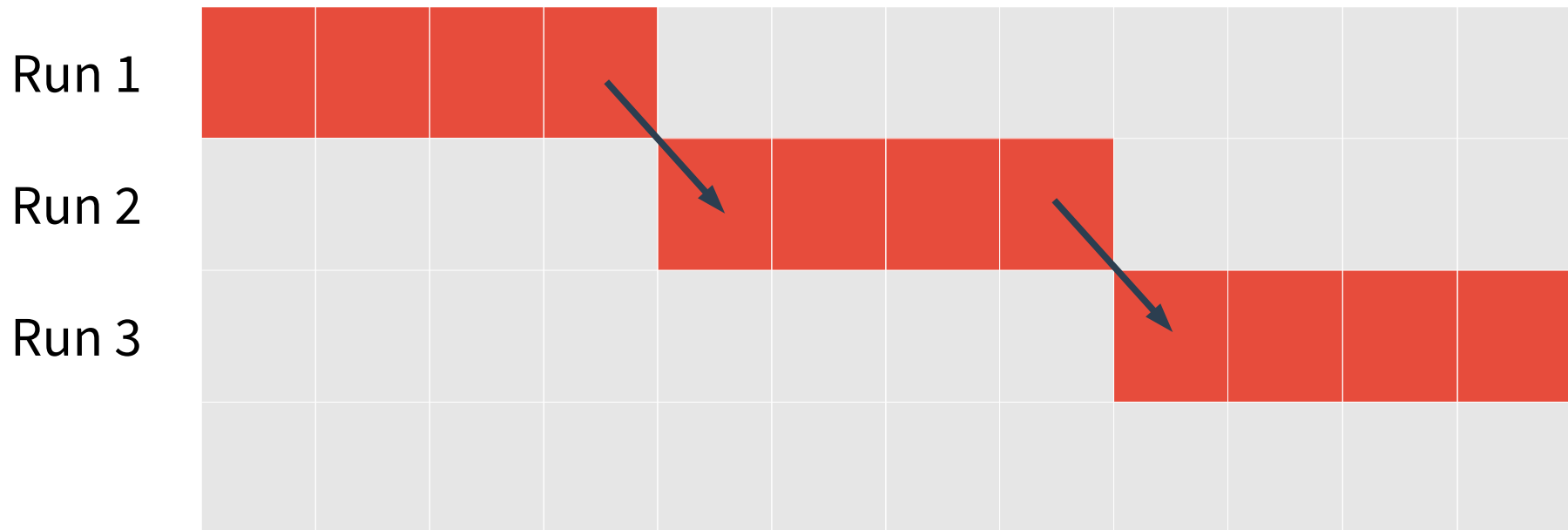
Flowchart of DiceMix



Communication Rounds (naive)



Communication Rounds (naive)



$4 + 4f$ rounds

Communication Rounds (DiceMix)

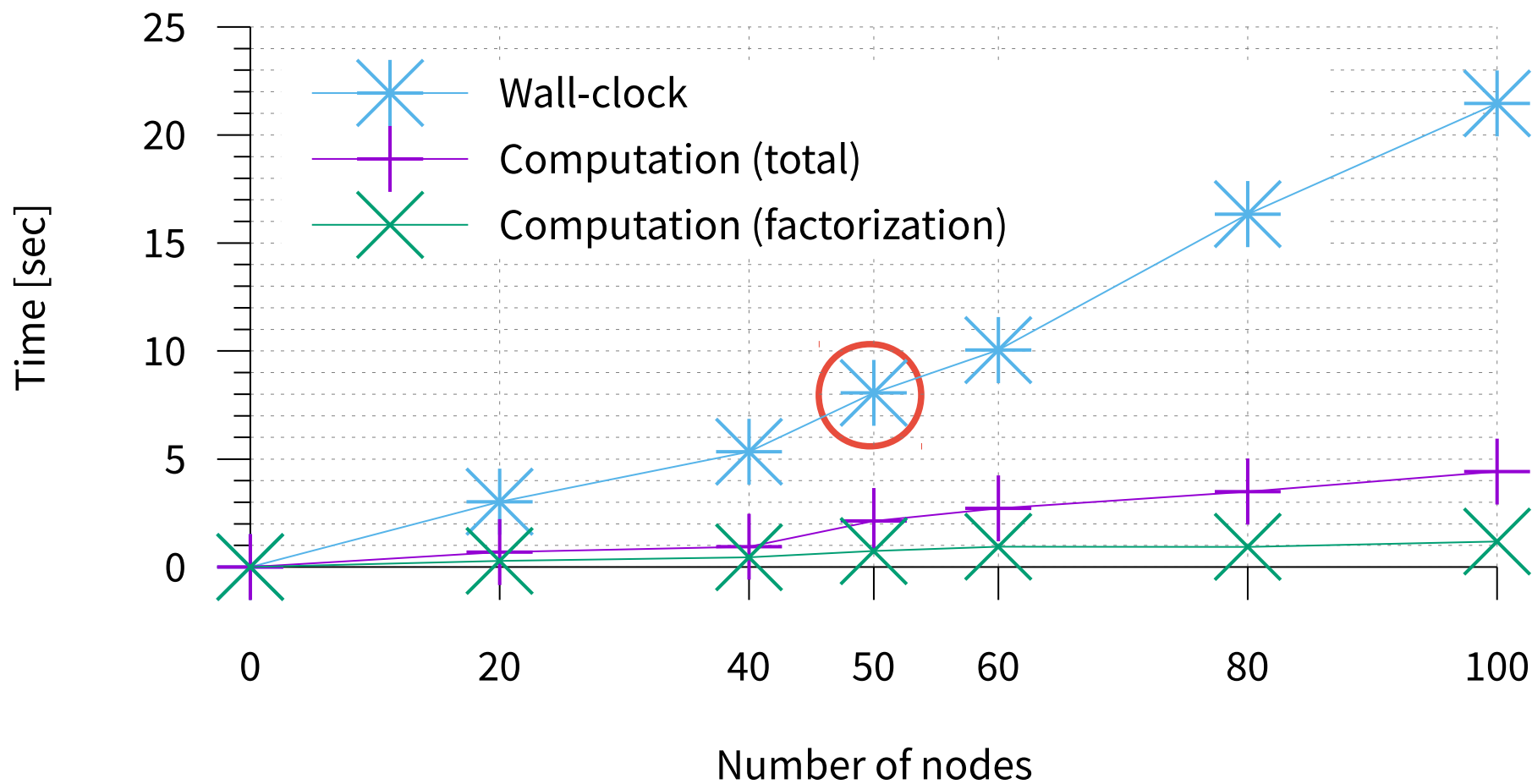


Communication Rounds (DiceMix)



$4 + 2f$ rounds







Performance



CoinShuffle++

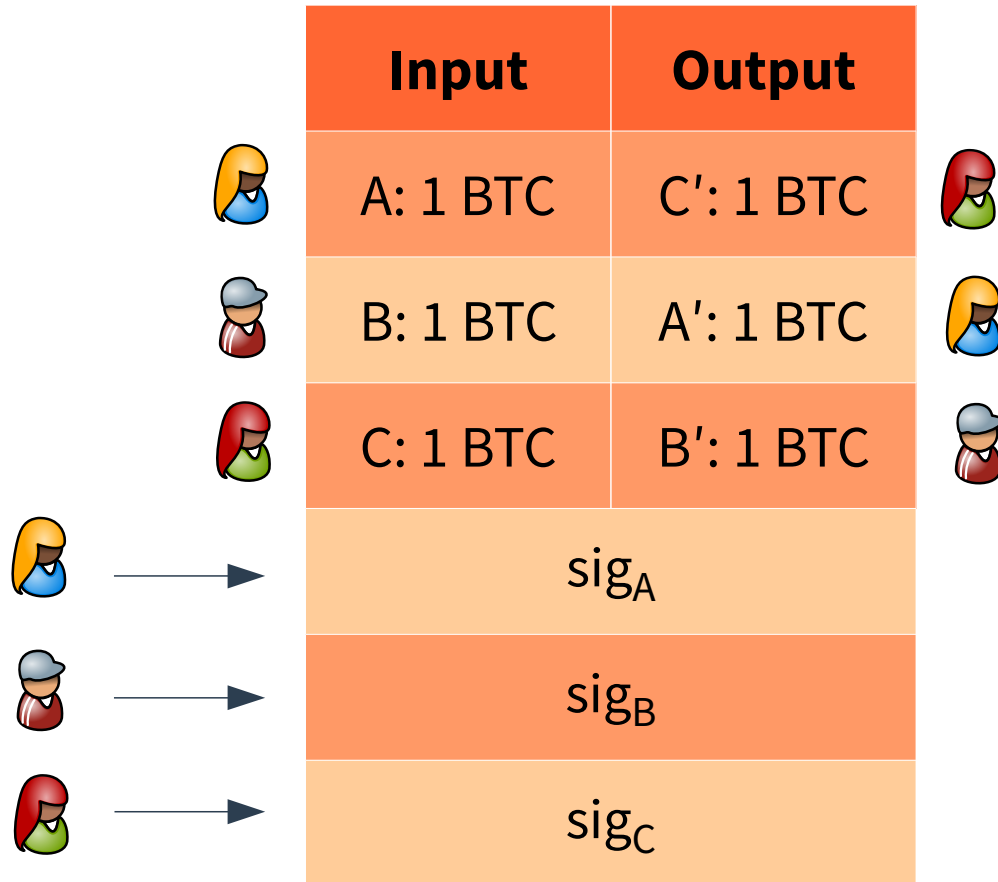
A P2P Coin Mixing Protocol based on DiceMix

Mixing without a Third Party (CoinJoin)

	Input	Output	
	A: 1 BTC	C': 1 BTC	
	B: 1 BTC	A': 1 BTC	
	C: 1 BTC	B': 1 BTC	

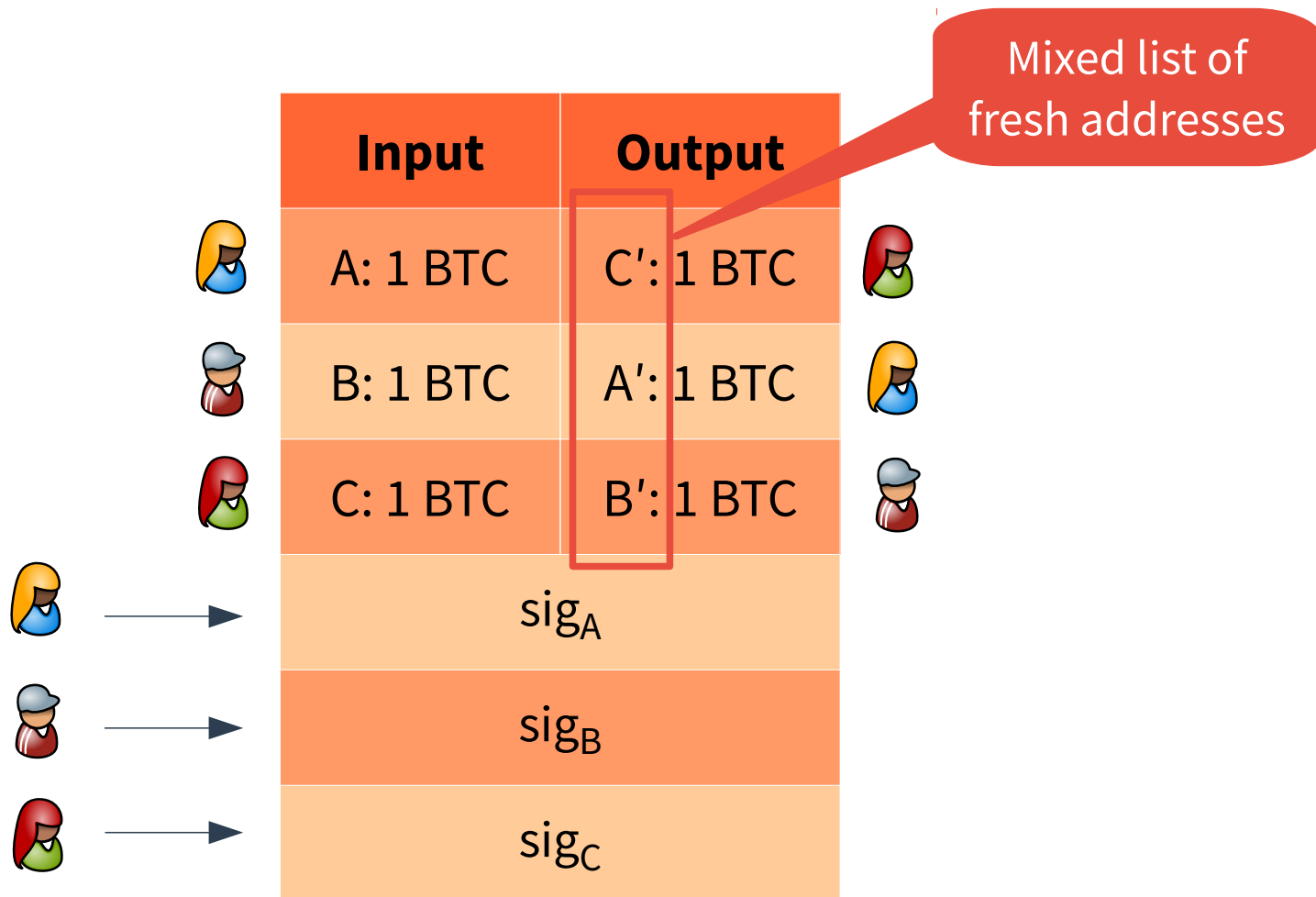
[CoinJoin, Maxwell 2013]

Mixing without a Third Party (CoinJoin)



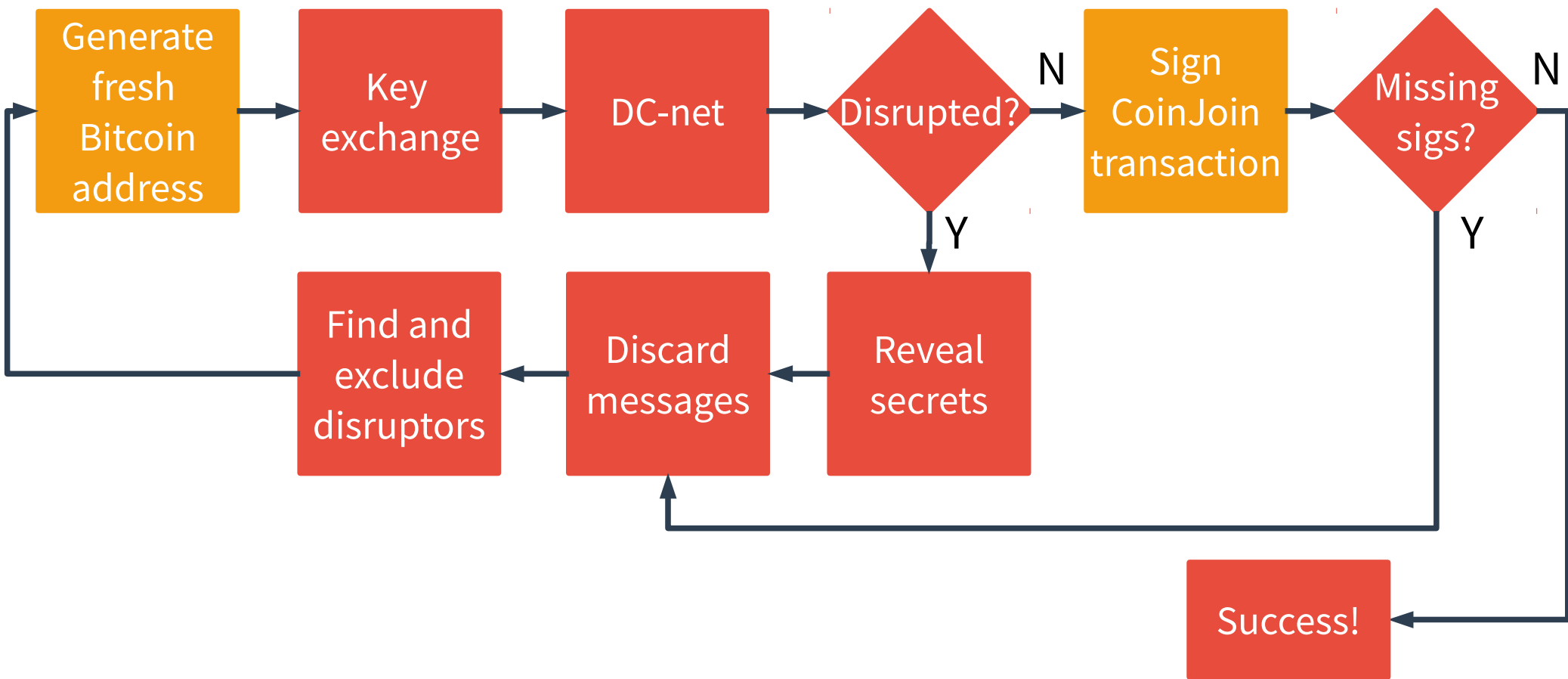
[CoinJoin, Maxwell 2013]

Mixing without a Third Party (CoinJoin)



[CoinJoin, Maxwell 2013]

Flowchart of CoinShuffle++



Comparison of CoinShuffle++ vs. TumbleBit

	TumbleBit (classic tumbler)	CoinShuffle++	
Anonymity set / payment	>> 100	~ 100	} off-chain
Bandwidth / payment	~ 420 bytes	~ 2250 bytes	
Total running time / payment	< 5 s	< 20 s	
Coordination required	no	yes	

Comparison of CoinShuffle++ vs. TumbleBit

	TumbleBit (classic tumbler)	CoinShuffle++	
Anonymity set / payment	>> 100	~ 100	} off-chain
Bandwidth / payment	~ 420 bytes	~ 2250 bytes	
Total running time / payment	< 5 s	< 20 s	
Coordination required	no	yes	
Sequential blocks / payment	2 + 1	1 + 1	} on-chain
Input-output pairs / payment	4 + 1	1 + 1	
Centralization	dedicated tumbler	P2P with bulletin board	
Collateral required	yes	no	

Comparison of CoinShuffle++ vs. TumbleBit

	TumbleBit (classic tumbler)	CoinShuffle++	
Anonymity set / payment	>> 100	~ 100	} off-chain
Bandwidth / payment	~ 420 bytes	~ 2250 bytes	
Total running time / payment	< 5 s	< 20 s	
Coordination required	no	yes	
Sequential blocks / payment	2 + 1	1 + 1	} on-chain
Input-output pairs / payment	4 + 1	1 + 1	
Centralization	dedicated tumbler	P2P with bulletin board	
Collateral required	yes	no	
DoS / Sybil protection	fees	performance penalty / fees	
Confidential Transactions	no	yes	



More Shuffling

Shameless Plugs

ValueShuffle


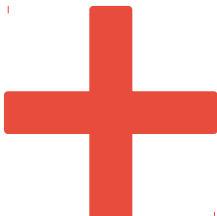
ValueShuffle = CoinShuffle++ + Confidential Transactions

ValueShuffle

ValueShuffle  CoinShuffle++  Confidential Transactions



- Hides amounts in transactions and provides anonymity

ValueShuffle

ValueShuffle  CoinShuffle++  Confidential Transactions



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

ValueShuffle

ValueShuffle  CoinShuffle++  Confidential Transactions


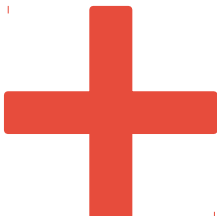
- Hides amounts in transactions and provides anonymity
- **Users with different amounts of money can mix!**
- User can mix and pay simultaneously in one transaction

ValueShuffle

ValueShuffle  CoinShuffle++  Confidential Transactions

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

ValueShuffle

ValueShuffle  CoinShuffle++  Confidential Transactions

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

PathShuffle

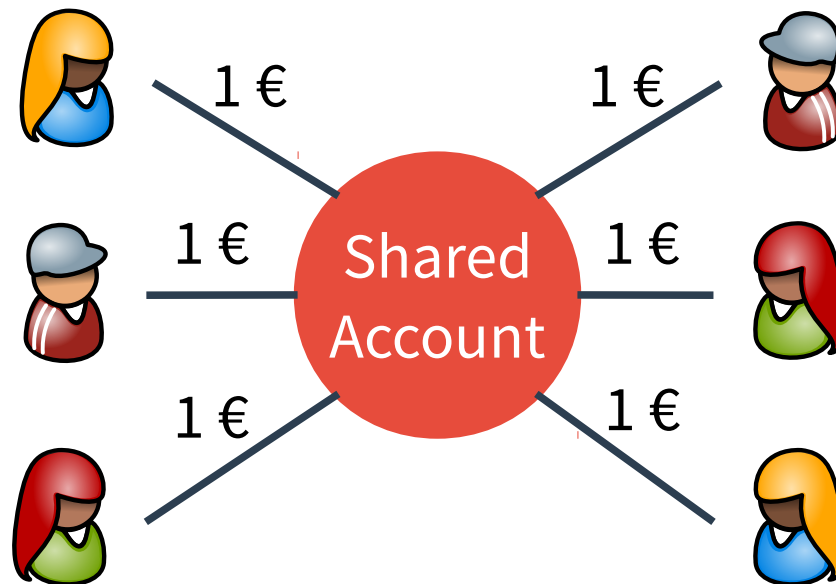
Money Mixing in Credit Networks

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

PathShuffle

Money Mixing in Credit Networks

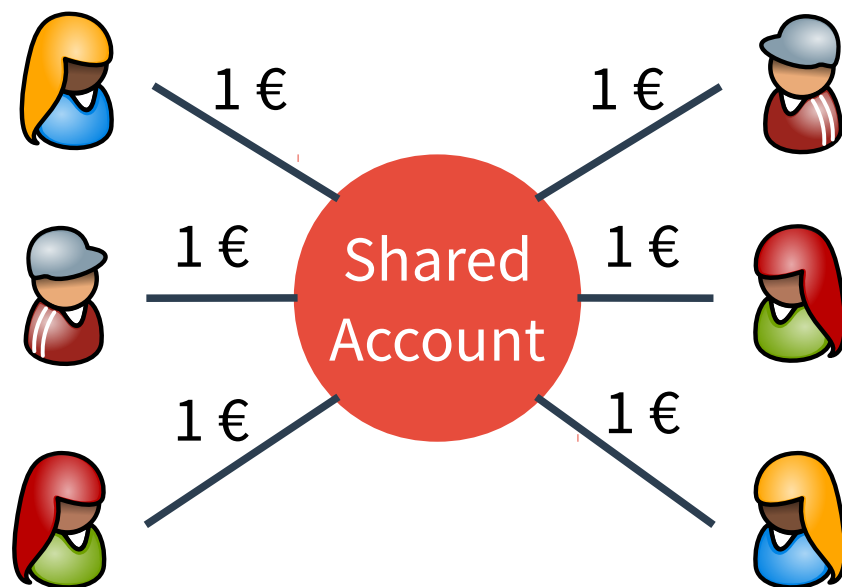
- Idea similar to CoinJoin, but there is no CoinJoin transaction
- Challenge: Simulate the CoinJoin via a shared account



PathShuffle

Money Mixing in Credit Networks

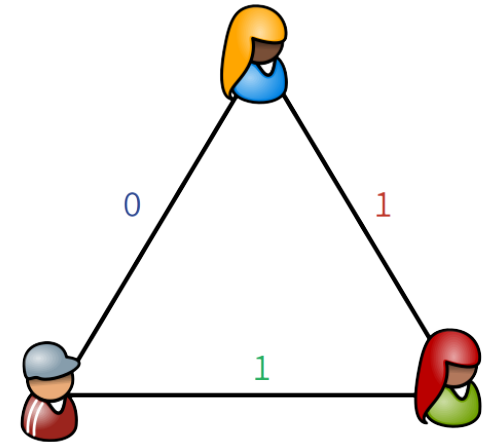
- Idea similar to CoinJoin, but there is no CoinJoin transaction
- Challenge: Simulate the CoinJoin via a shared account



Accepted by PoPETs 2017

Take-Home Message

- DC-nets are practical
 - No honest majority necessary
 - Only simple crypto, simple protocol
 - Only $4 + 2f$ rounds
- P2P coin mixing is practical
 - No central party necessary
 - CoinShuffle++ is an efficient solution

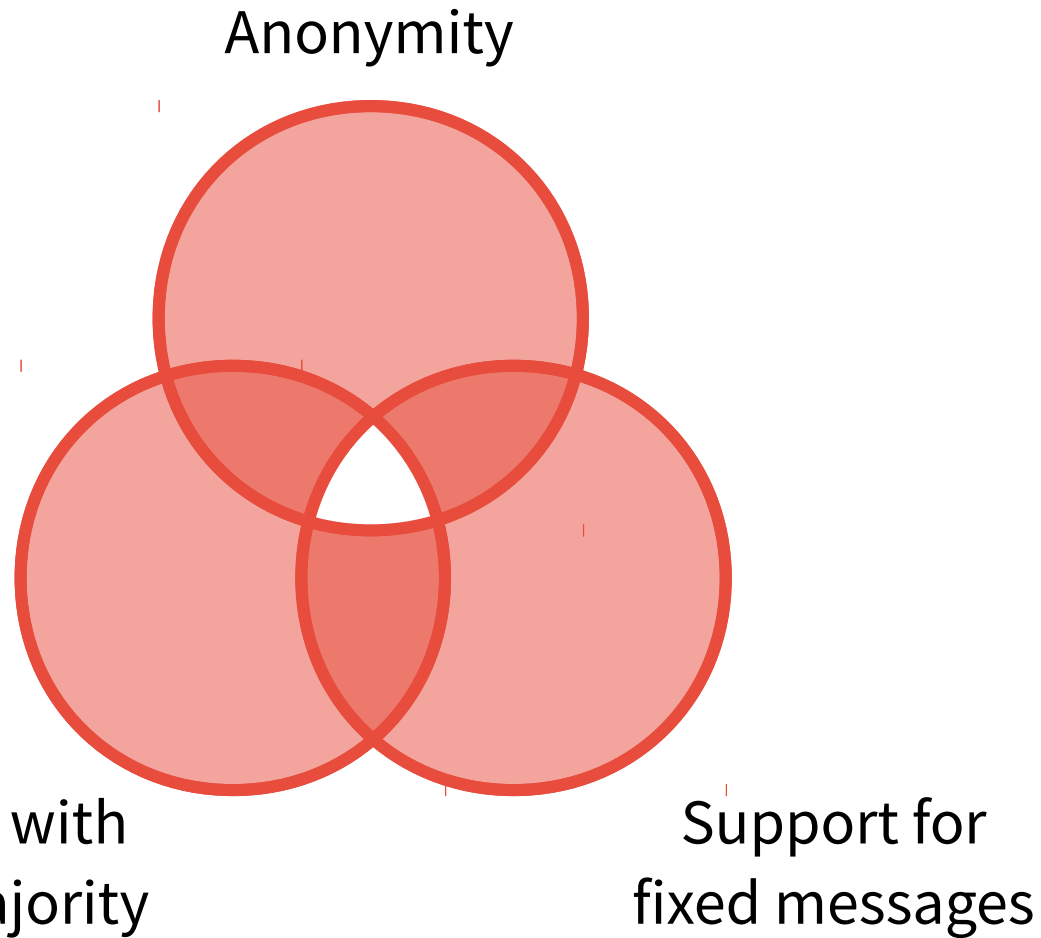


Work in progress:

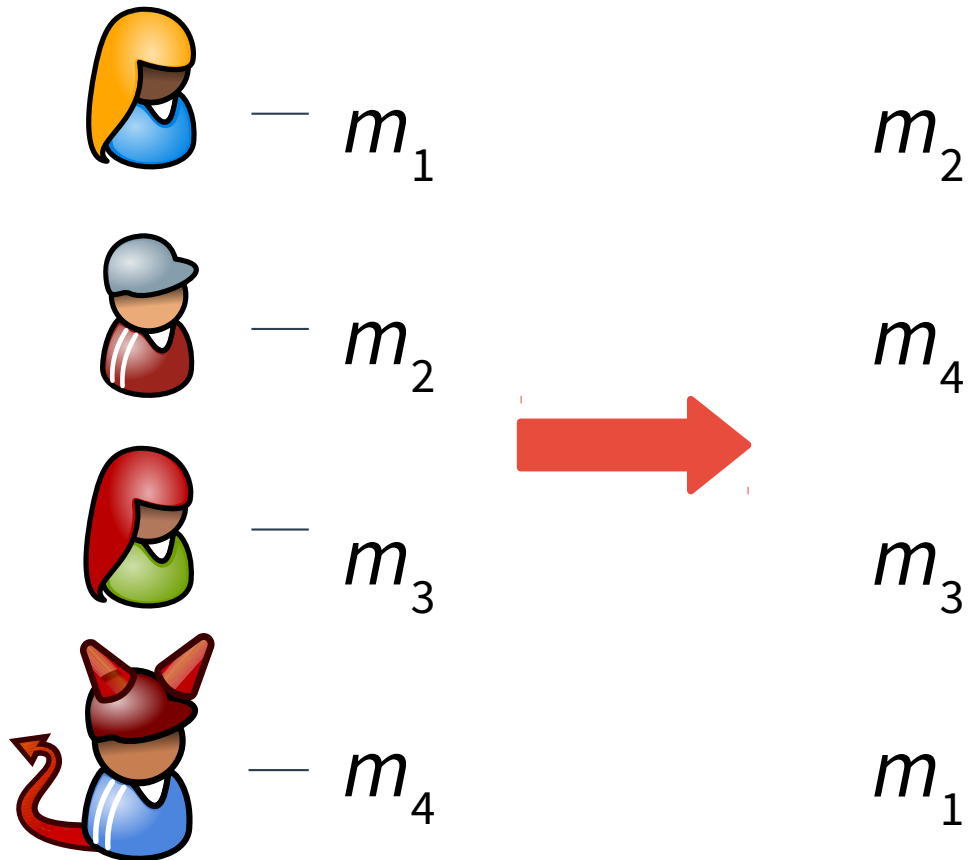
<https://github.com/real-or-random/python-dicemix>

Backup Slides

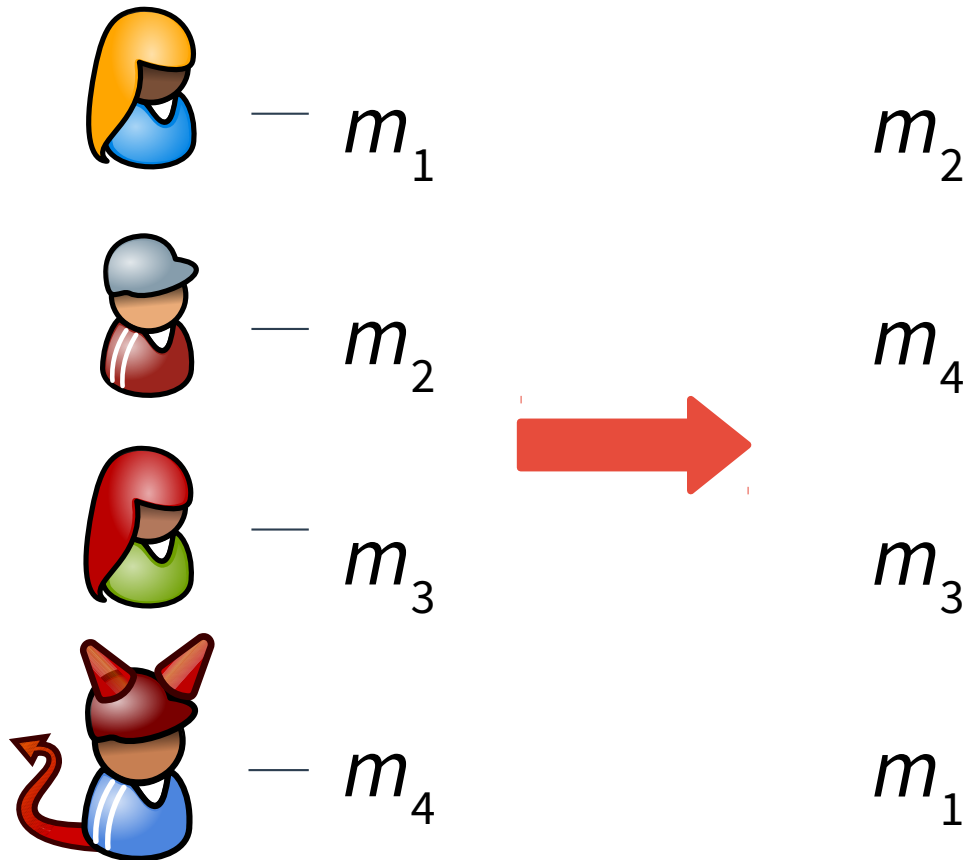
Freshness Is Necessary



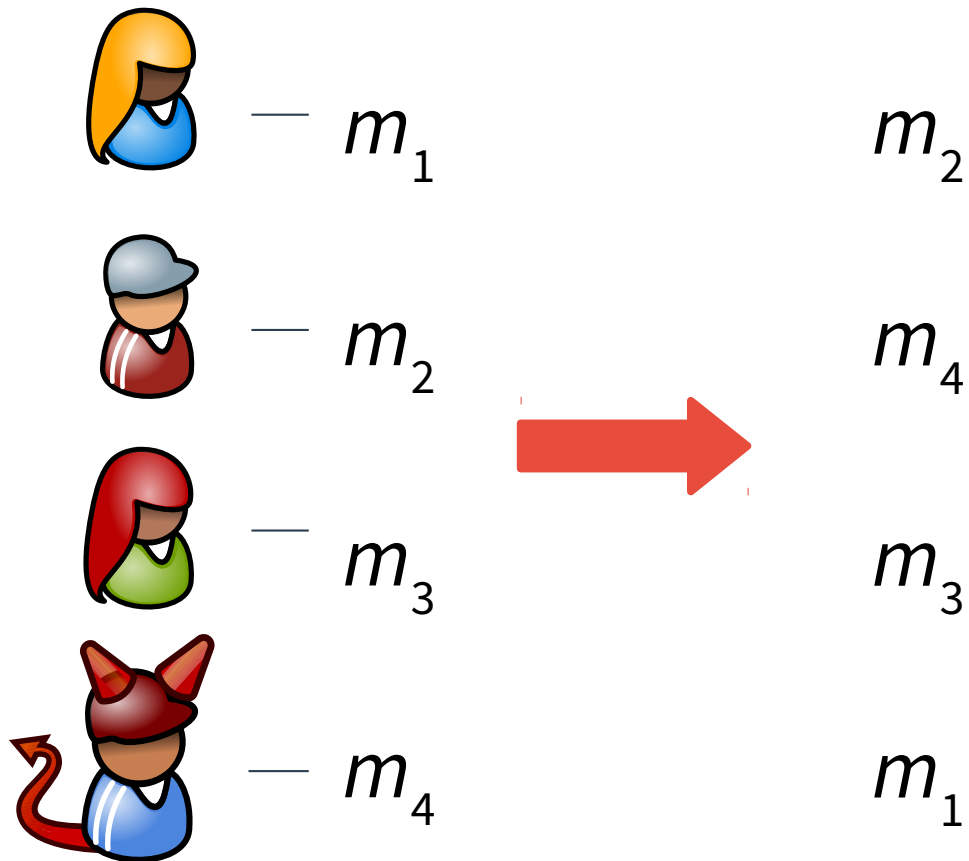
Idea of Attack on Anonymity



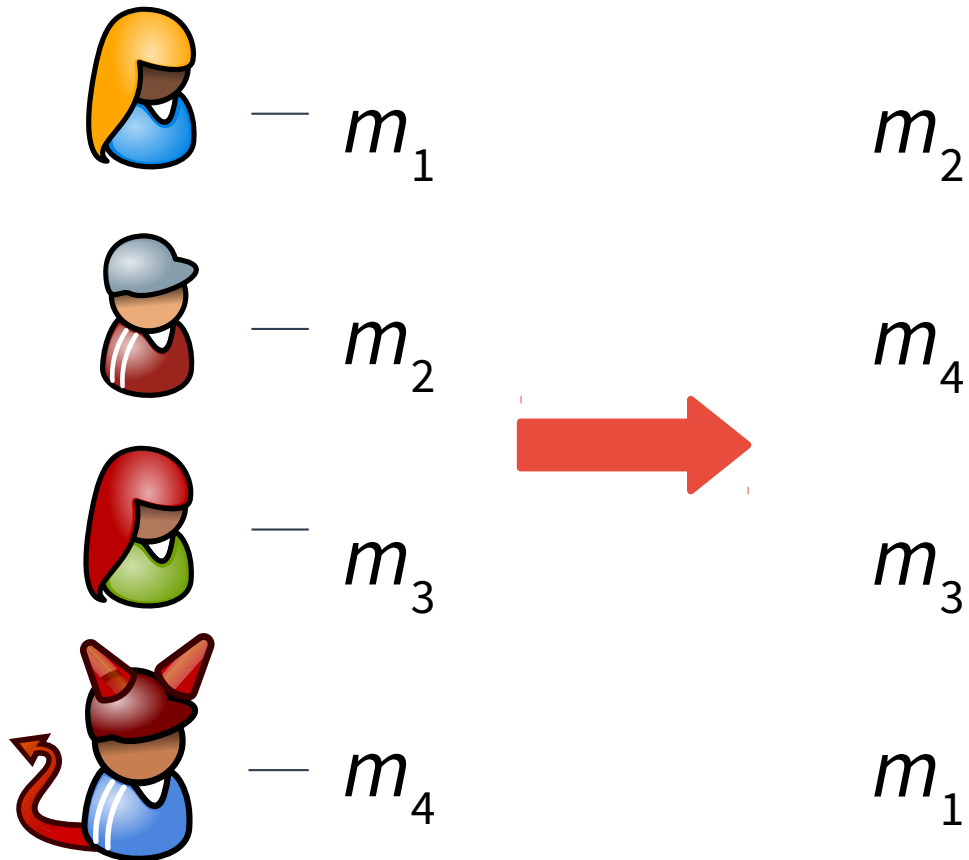
Idea of Attack on Anonymity



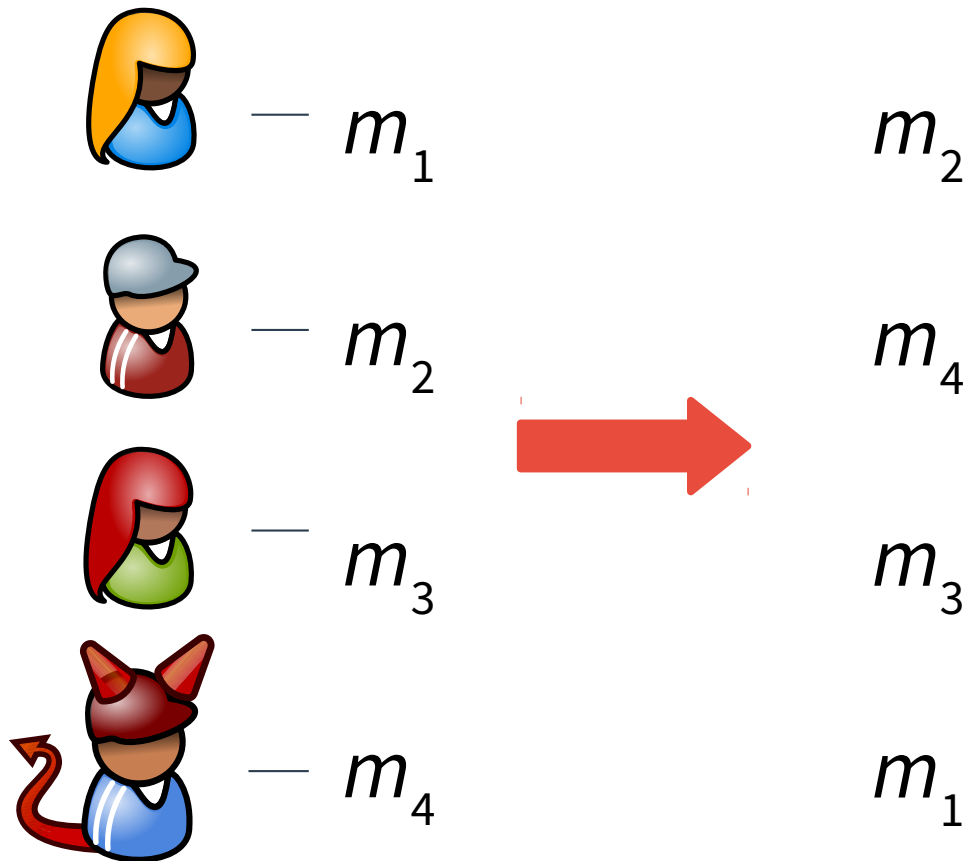
Idea of Attack on Anonymity



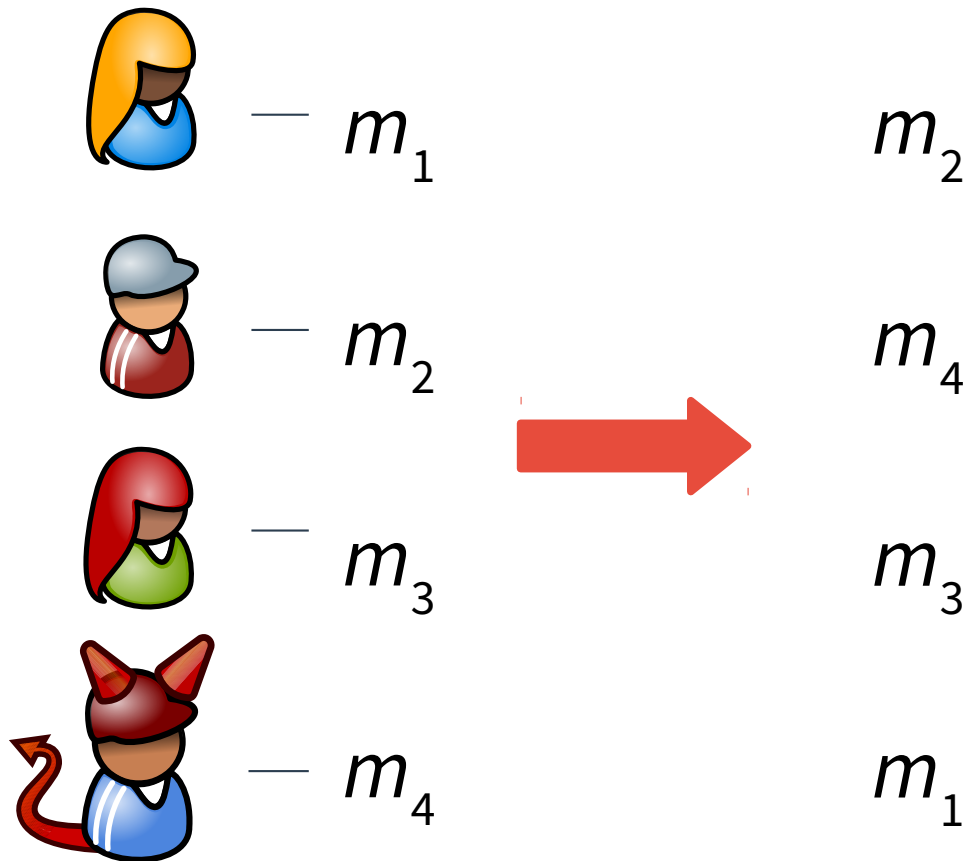
Idea of Attack on Anonymity



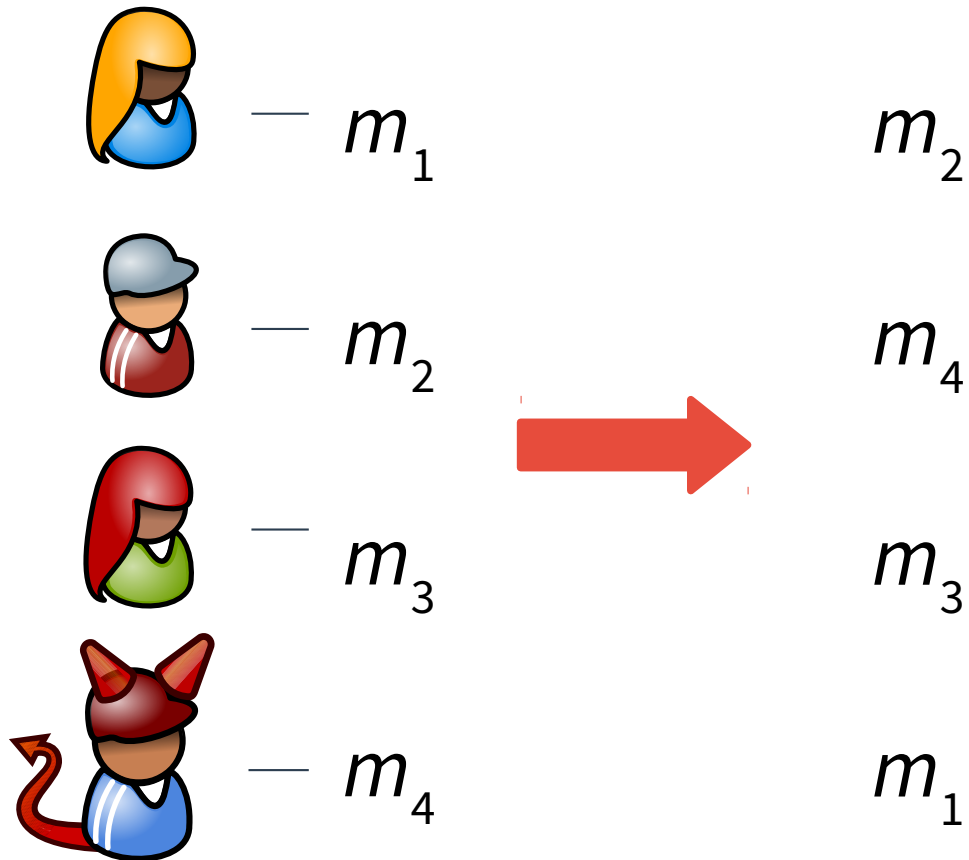
Idea of Attack on Anonymity



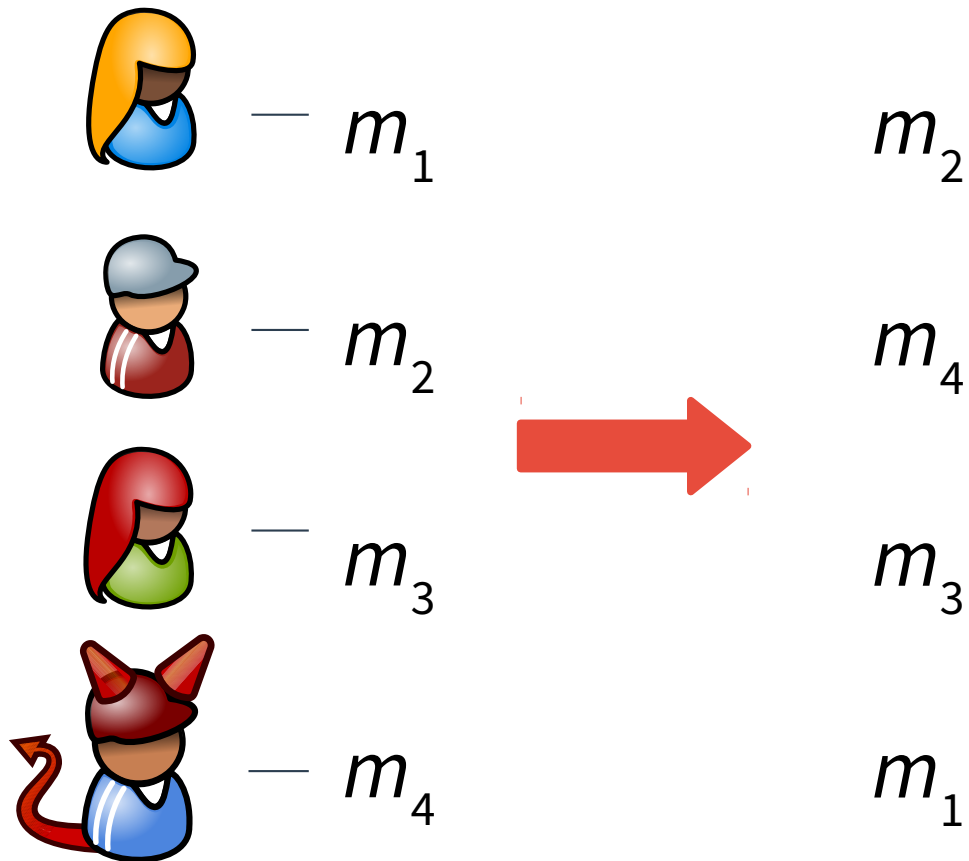
Idea of Attack on Anonymity



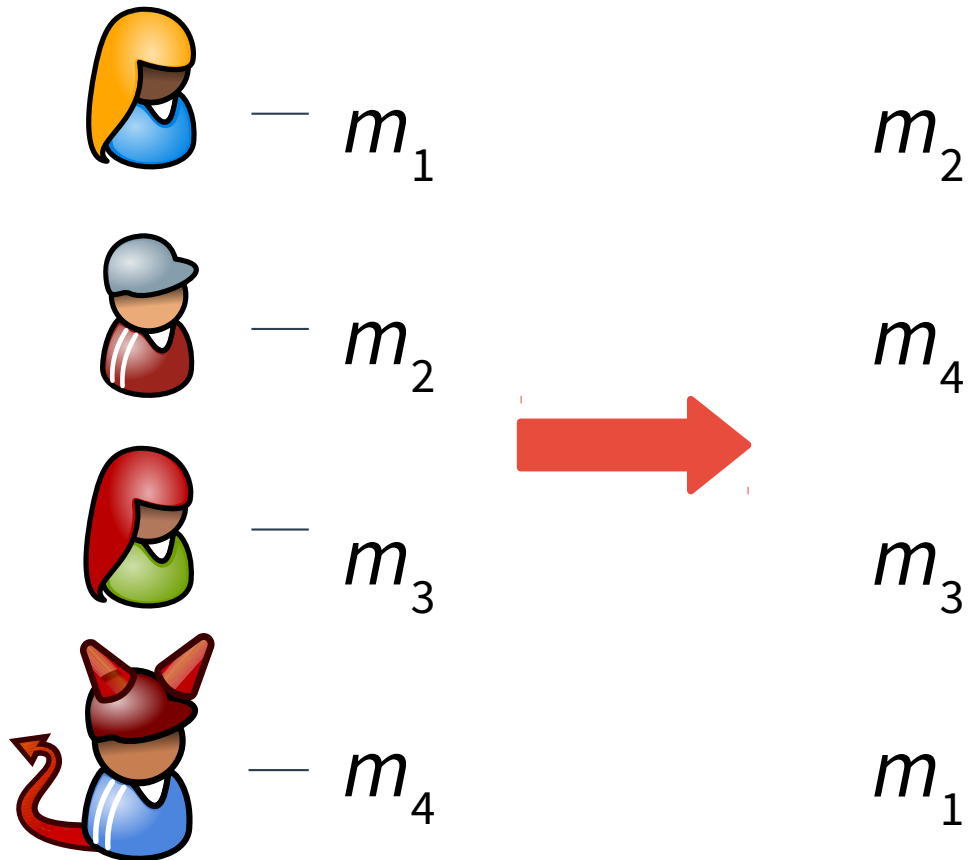
Idea of Attack on Anonymity



Idea of Attack on Anonymity



Idea of Attack on Anonymity



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)												

$4 + 2f$ rounds

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)

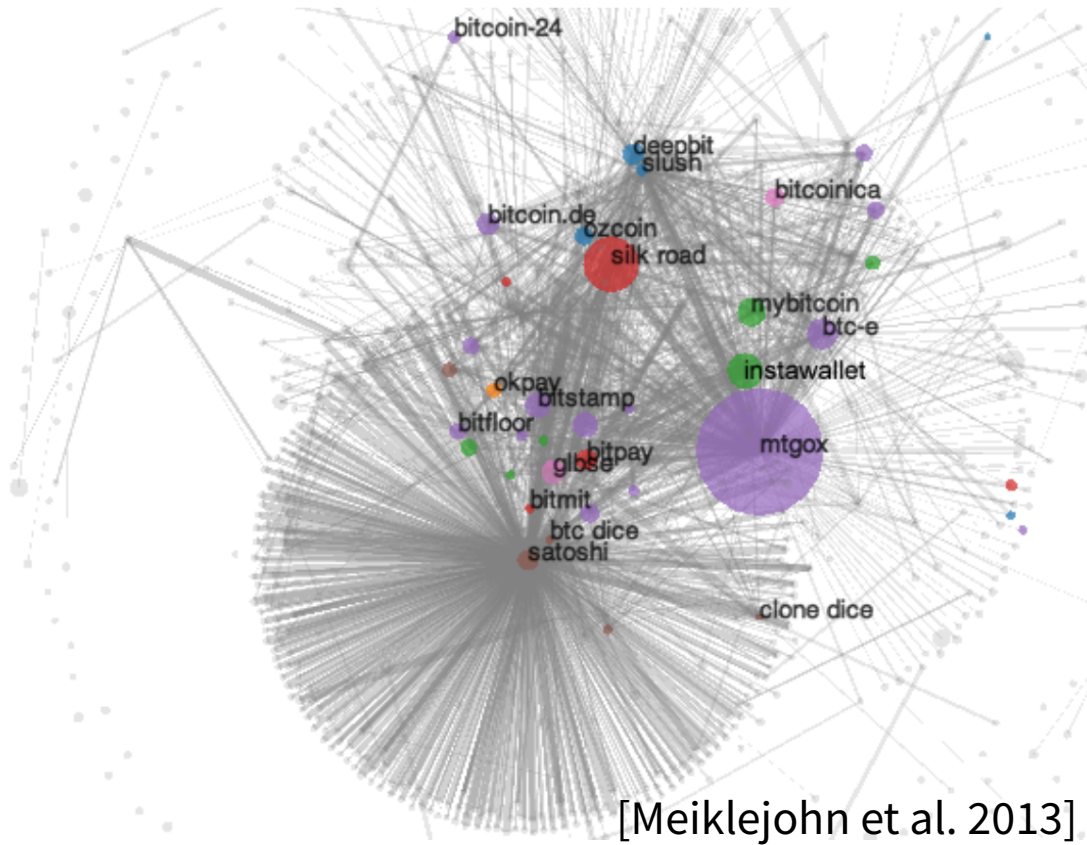
Run 1	KE	CM	DC	SK								
Run 2			KE	CM	DC RV							
Run 3												
(Run 4)												

$4 + 2f$ rounds

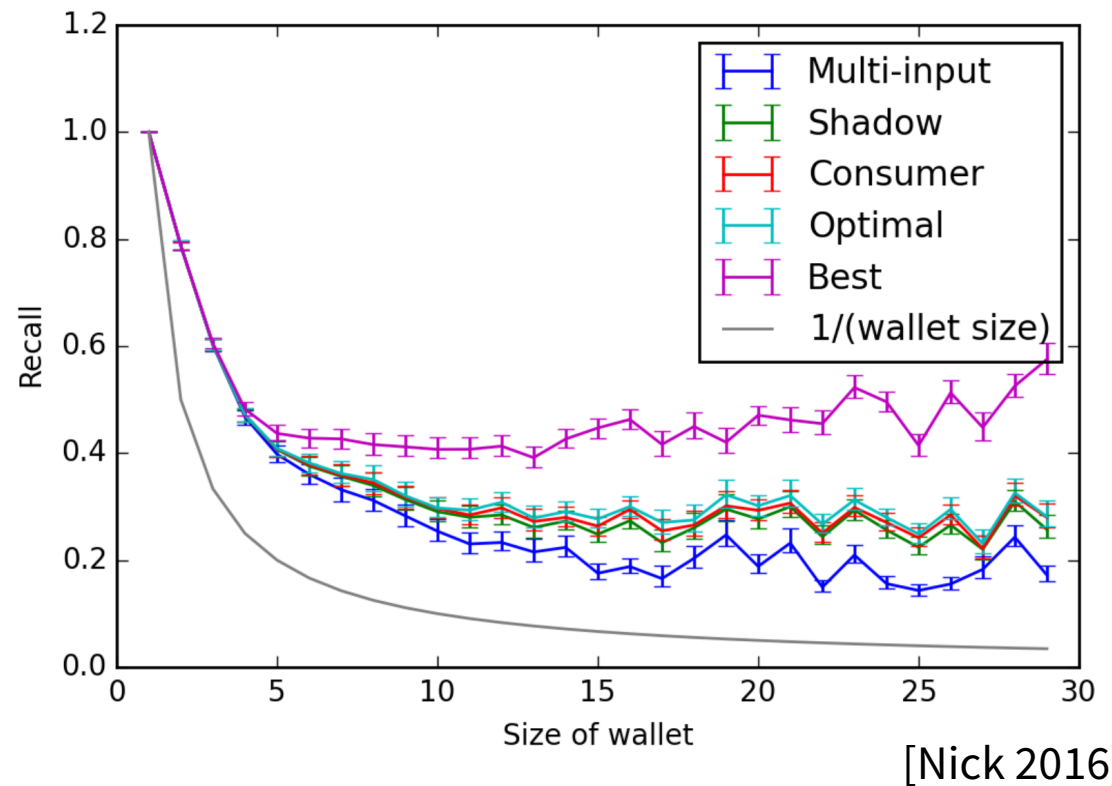
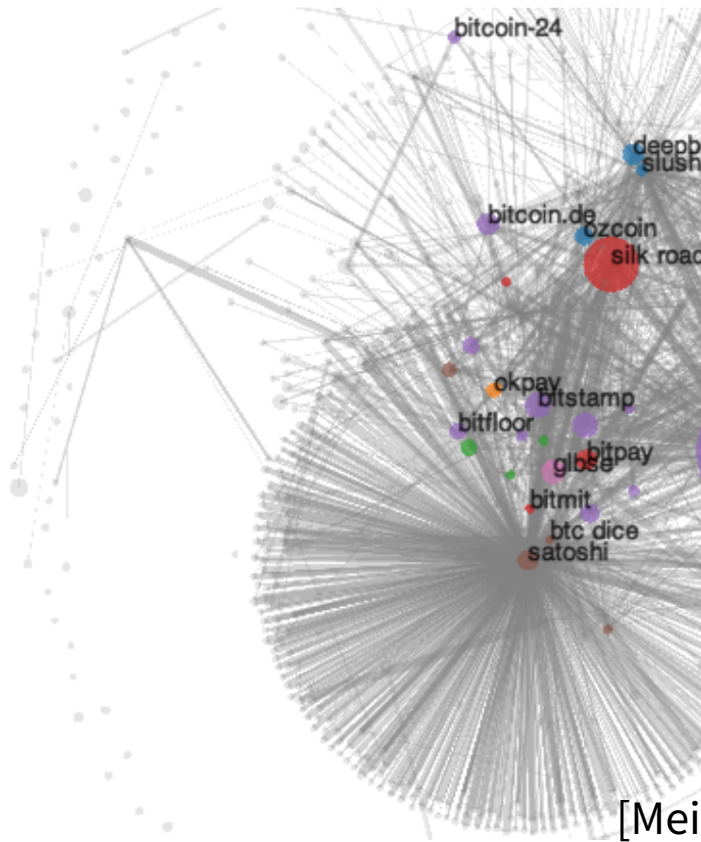
Transaction Linkability

A Threat to Privacy

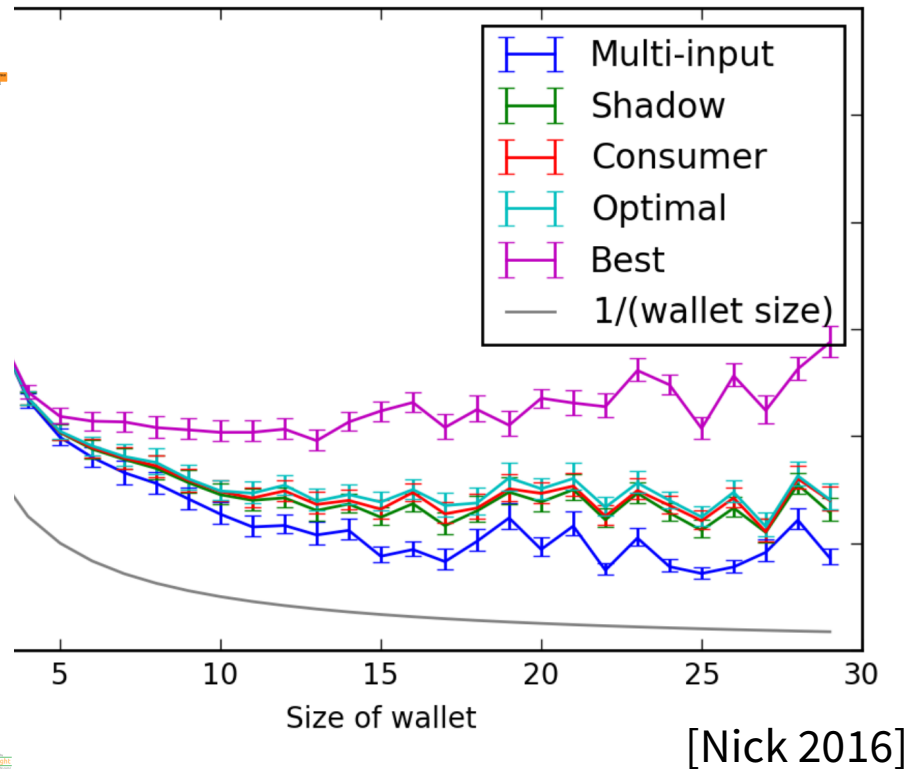
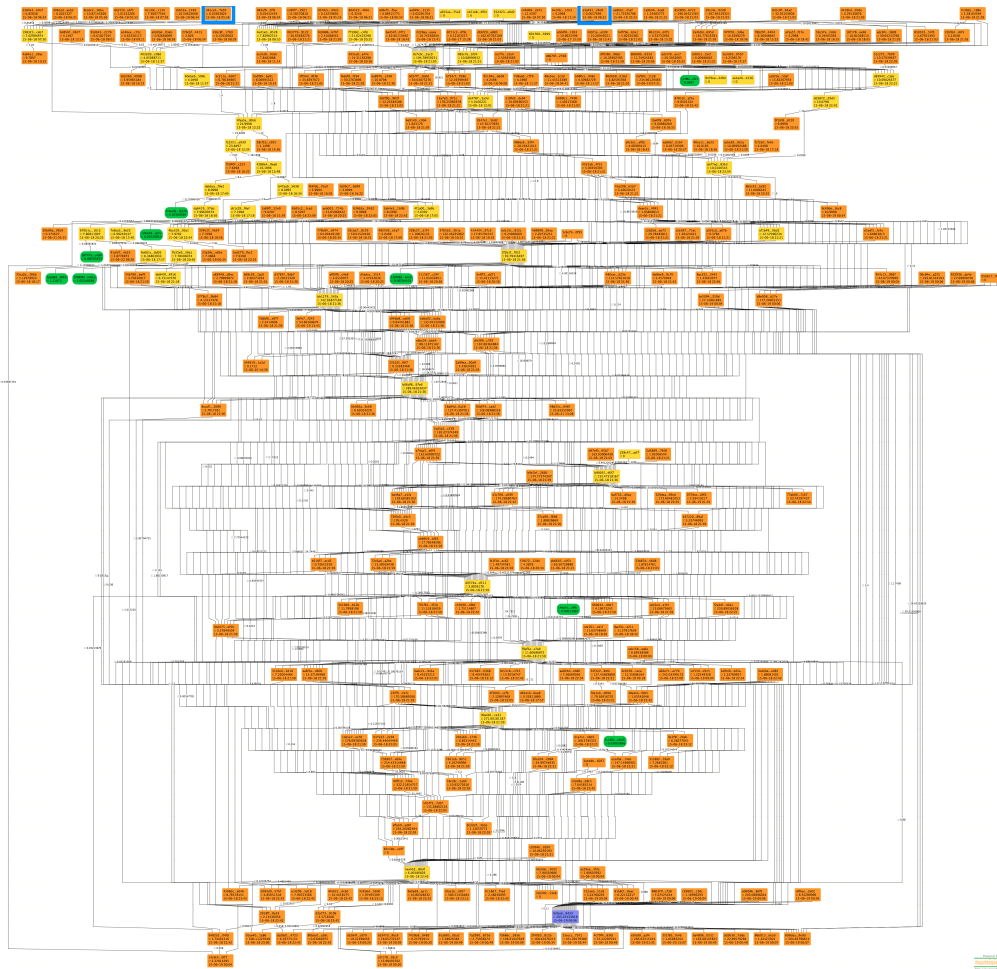
Linkability and Deanononymization Attacks



Linkability and Deanononymization Attacks

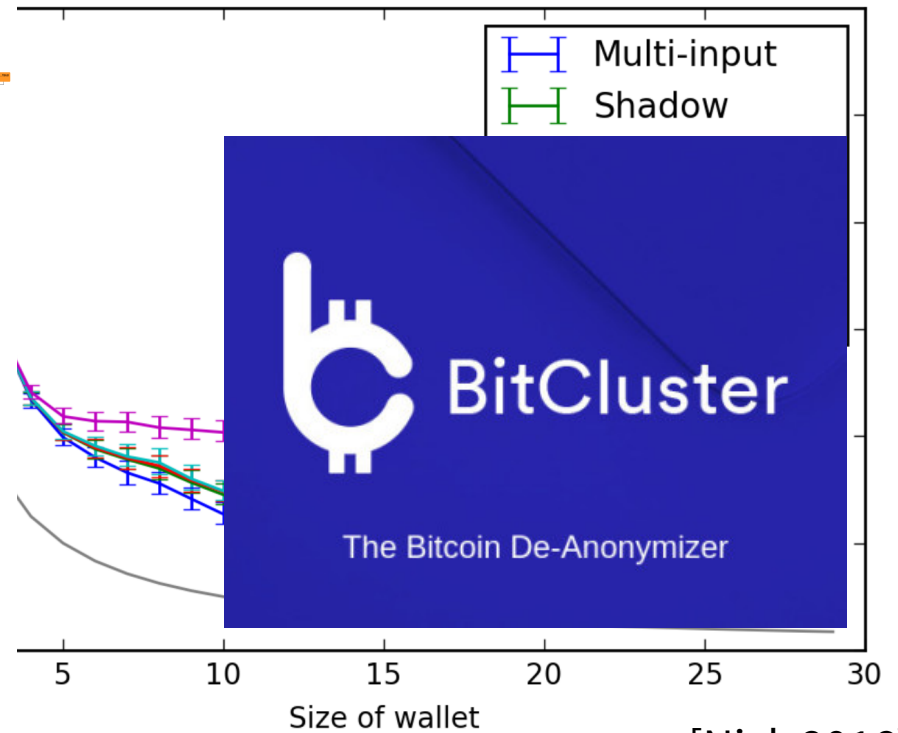
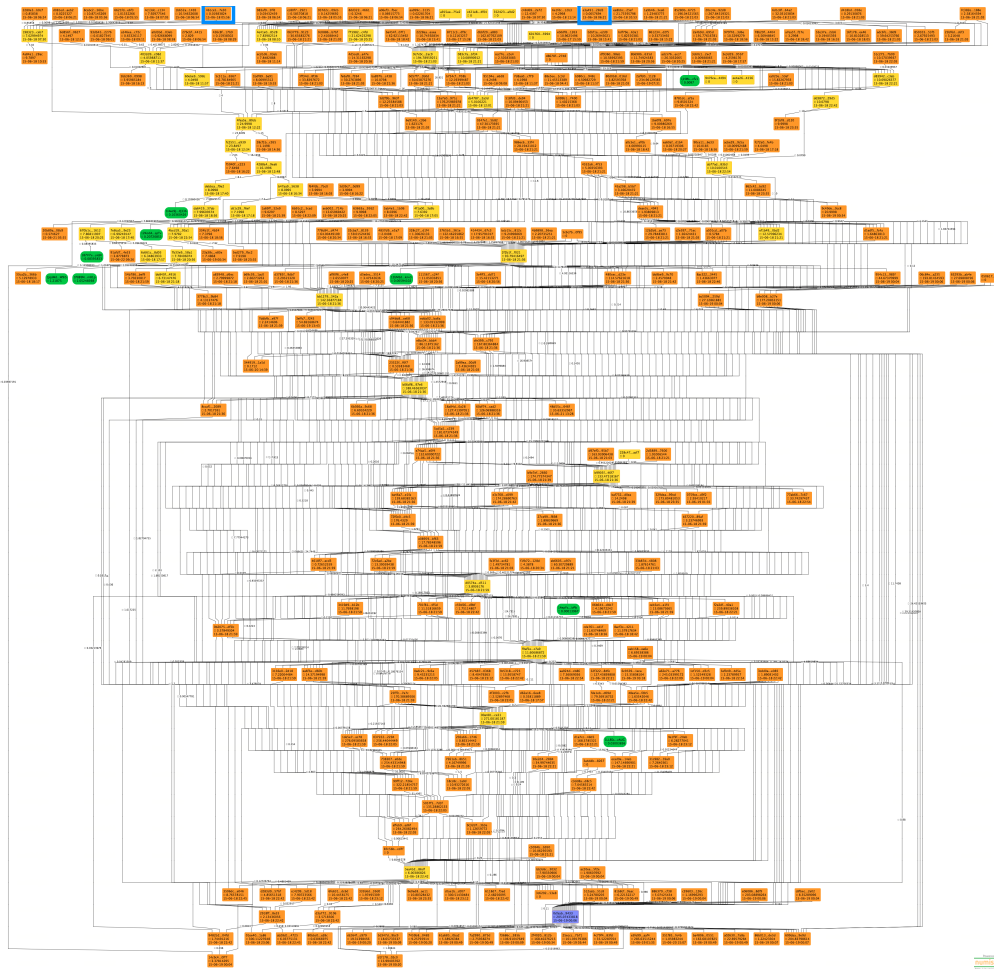


Linkability and Deanonimization Attacks



Bitlodine [Spagnuolo, Maggi, Zanero 2013]

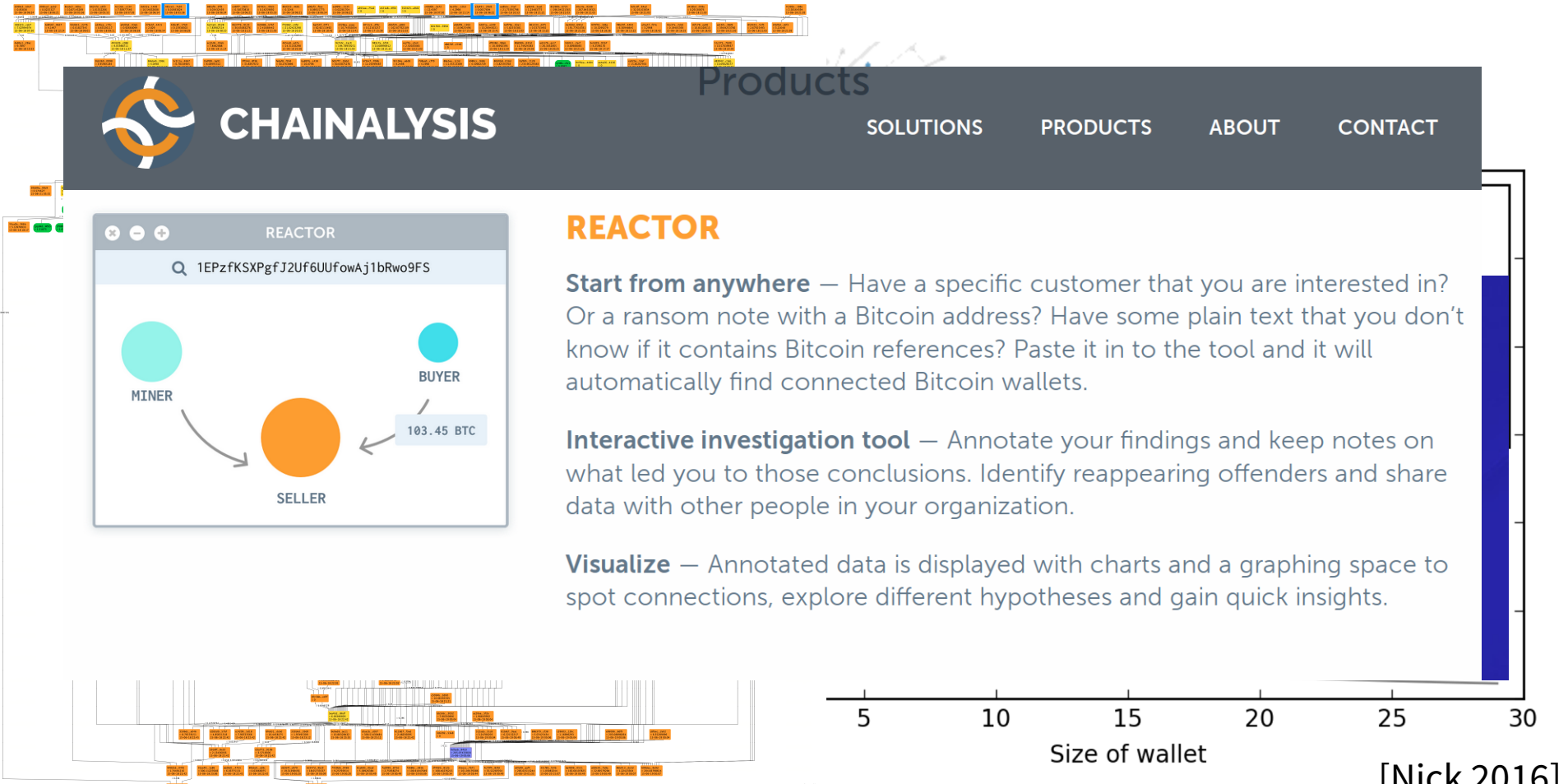
Linkability and Deanonimization Attacks



[Nick 2016]

Bitlodine [Spagnuolo, Maggi, Zanero 2013]

Linkability and Deanonimization Attacks



The image shows a screenshot of the Chainalysis Reactor web interface. At the top, there is a navigation bar with the Chainalysis logo and the word "Products" in a large, semi-transparent font. The navigation menu includes "SOLUTIONS", "PRODUCTS", "ABOUT", and "CONTACT". Below the navigation bar, a search bar contains the Bitcoin address "1EPzfKSXPgfJ2Uf6UUfowAj1bRwo9FS". The main content area displays a transaction flow diagram with three nodes: "MINER" (a teal circle), "SELLER" (an orange circle), and "BUYER" (a blue circle). An arrow points from the MINER to the SELLER, and another arrow points from the SELLER to the BUYER. A transaction box next to the BUYER node shows "103.45 BTC". To the right of the interface, there is a vertical blue bar and a horizontal axis labeled "Size of wallet" with tick marks at 5, 10, 15, 20, 25, and 30. The text "[Nick 2016]" is located at the bottom right of this section.

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]

Bitlodine [Spagnuolo, Maggi, Zanero 2013]

Performance

- We use an untrusted bulletin board, e.g., IRC server, but just for communication.
- CoinShuffle++ terminates in $4 + 2f$ rounds with f disruptive users
 - **< 10 seconds** to create CoinJoin transaction with 50 honest users (unoptimized)
 - old CoinShuffle: about 3 min
- Work in progress:
<https://github.com/real-or-random/python-dicemix>

DC-nets in Practice

DC-nets in Practice

- Key exchange to establish shared keys

DC-nets in Practice

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

DC-nets in Practice

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

DC-nets in Practice

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

