

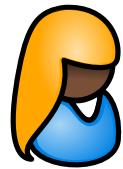
# P2P Mixing and Unlinkable Bitcoin Transactions

**Tim Ruffing**  
@real\_or\_random

**Pedro Moreno-Sánchez**  
@pedrorechez

**Aniket Kate**  
@aniketpkate

# P2P Mixing



— A



— B

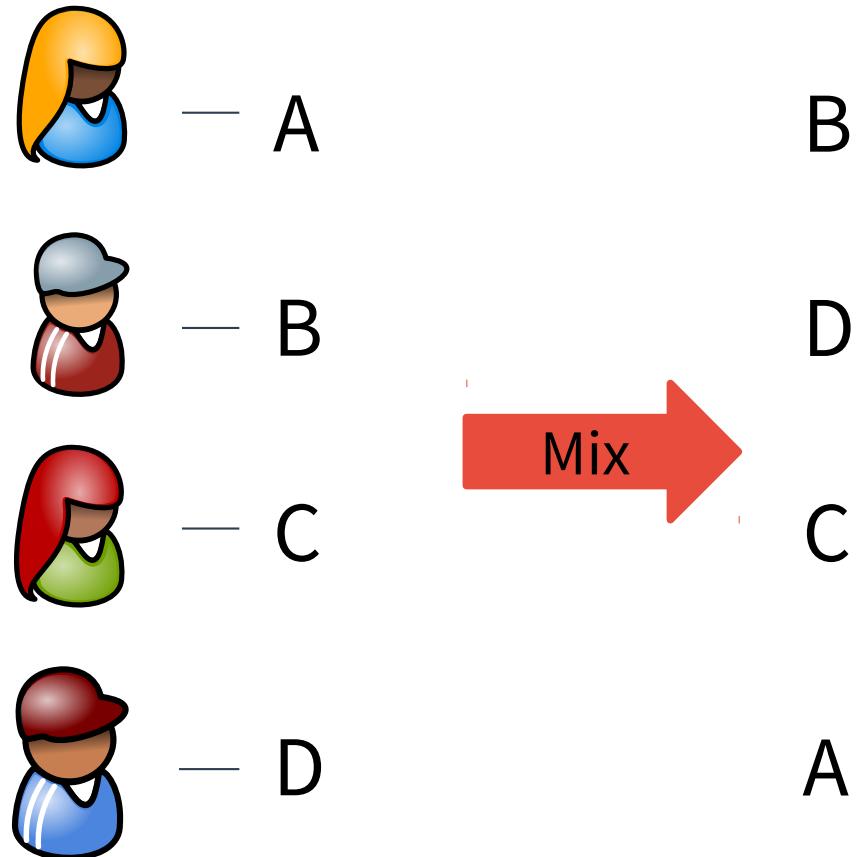


— C

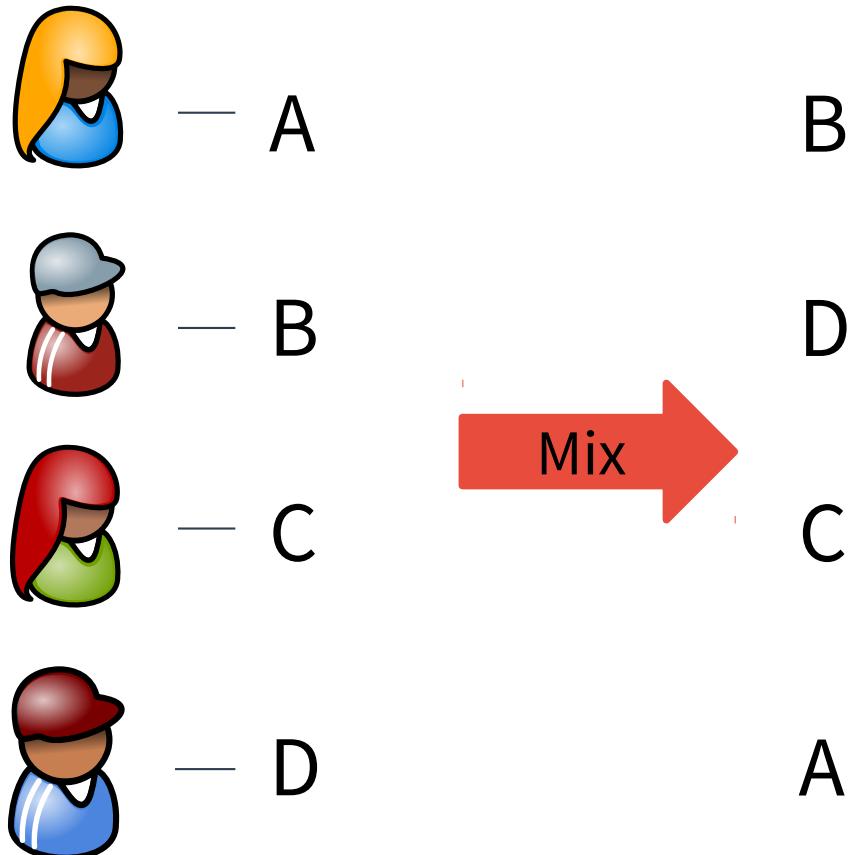


— D

# P2P Mixing



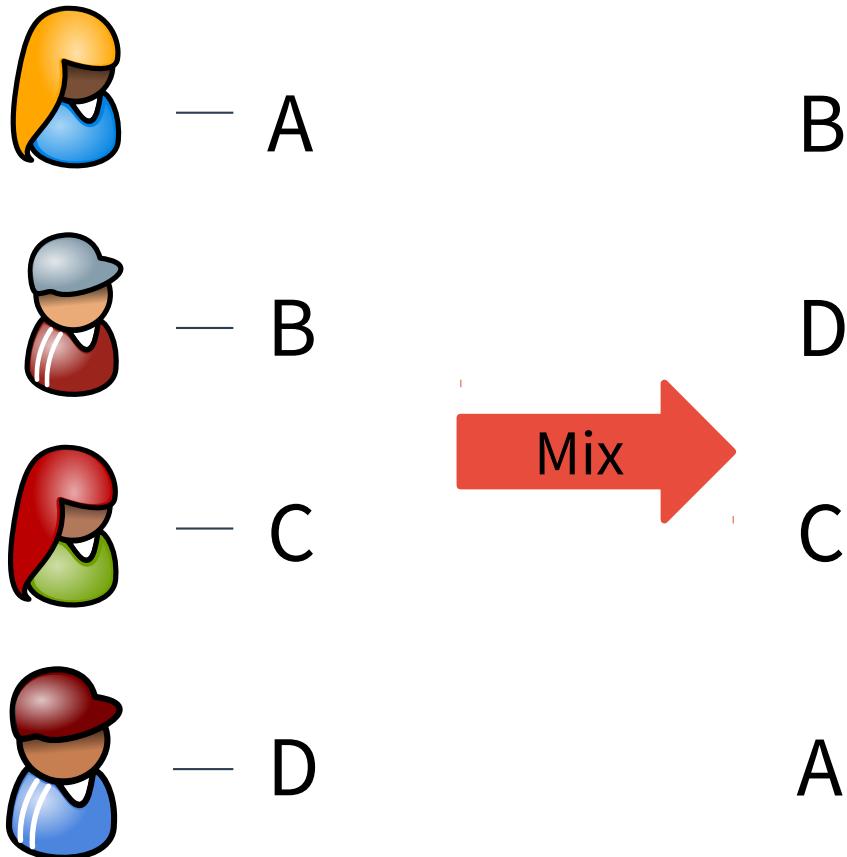
# P2P Mixing



## Confirmation

- Peers agree on the output and confirm it

# P2P Mixing



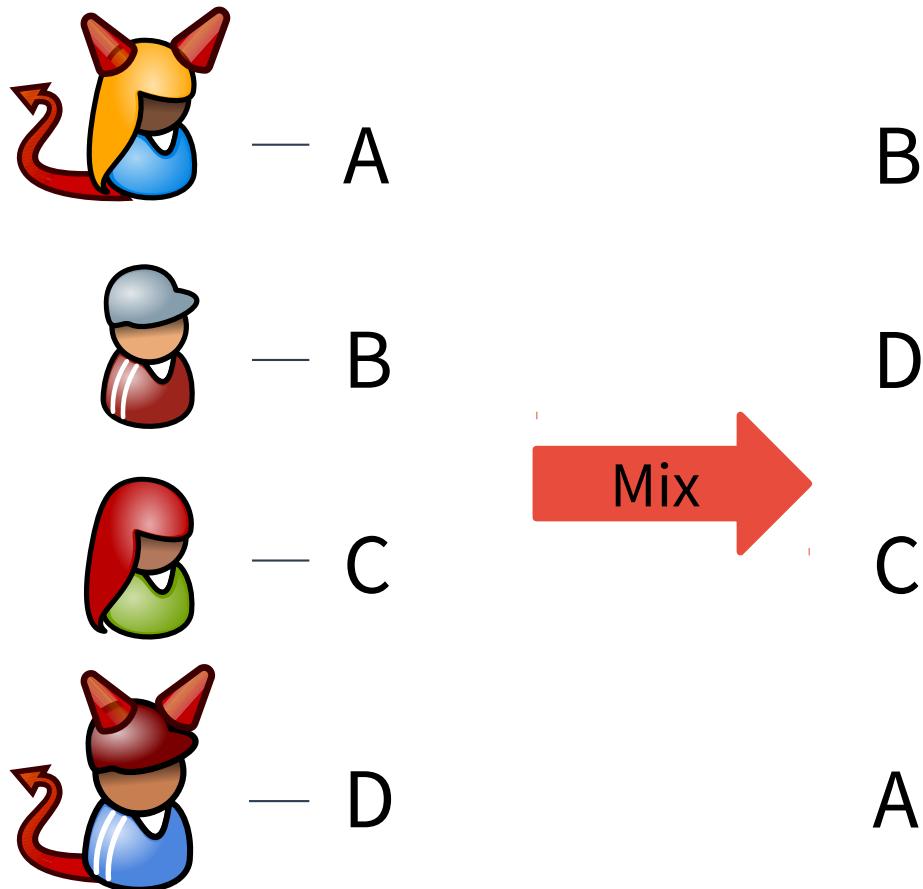
## Confirmation

- Peers agree on the output and confirm it

## P2P Trust model

- No mutual trust, no third-party routers

# P2P Mixing



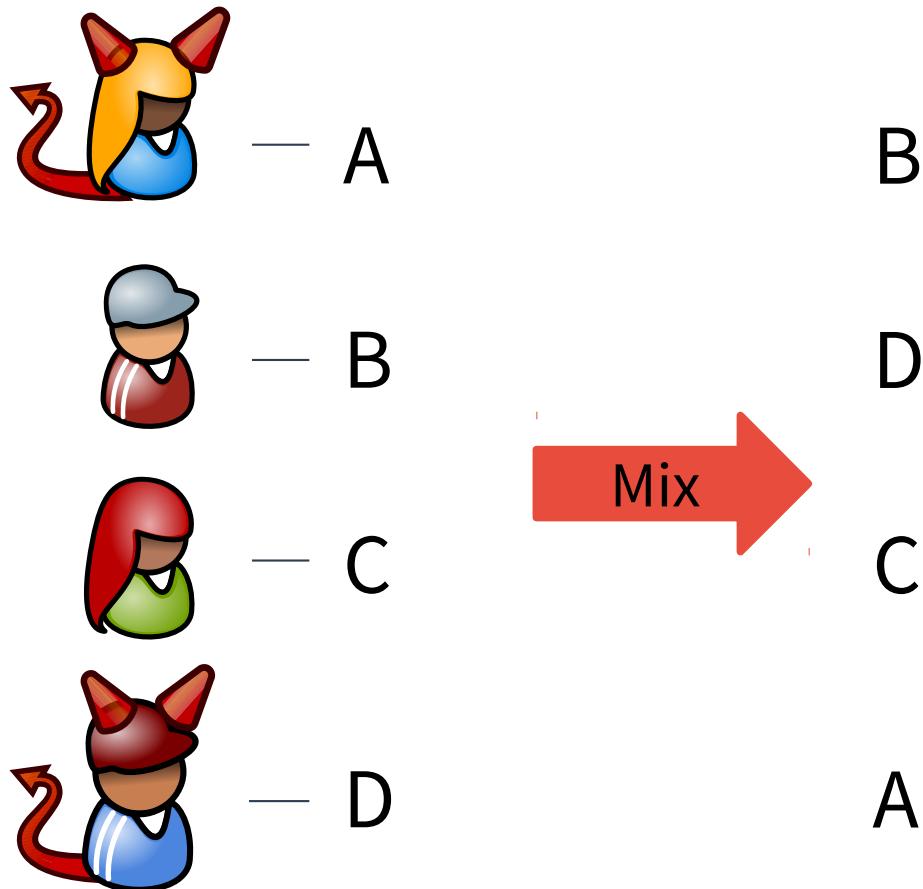
## Confirmation

- Peers agree on the output and confirm it

## P2P Trust model

- No mutual trust, no third-party routers

# P2P Mixing



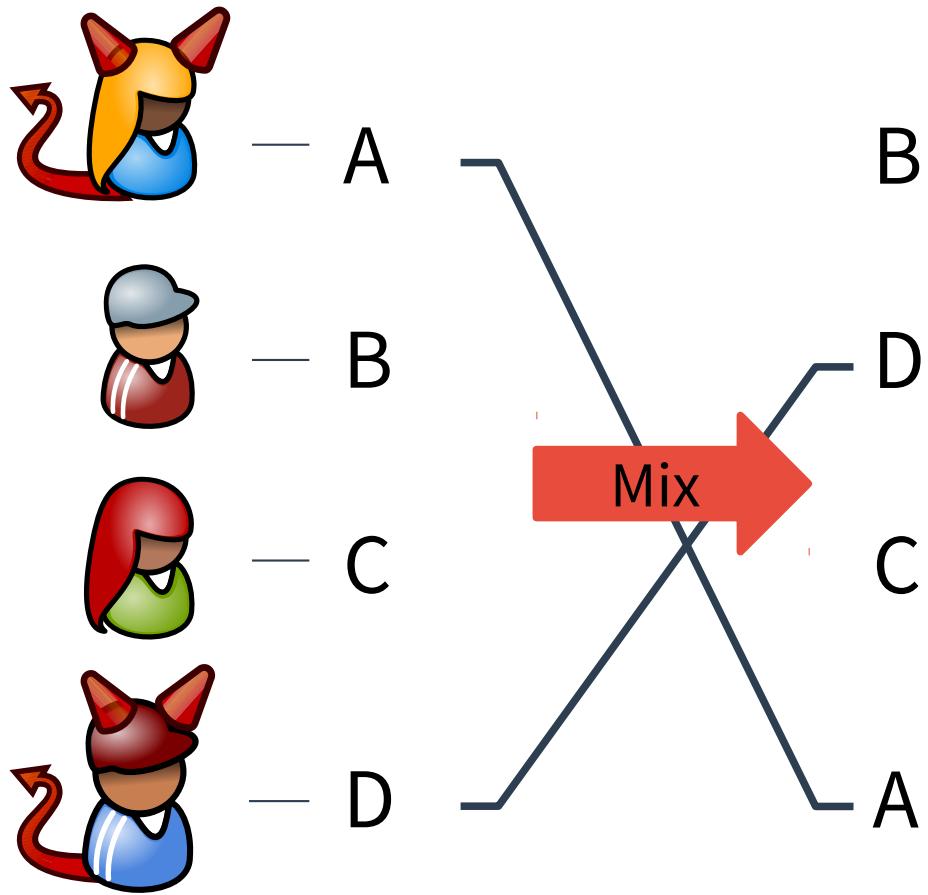
## Confirmation

- 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



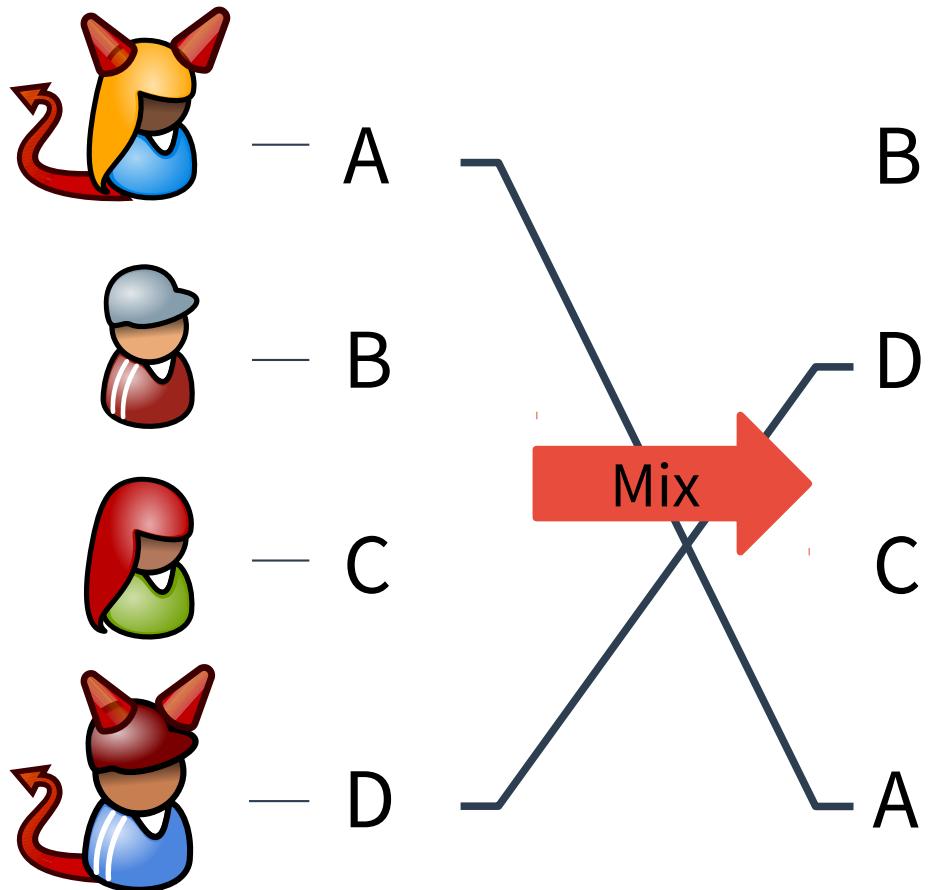
## Confirmation

- 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)

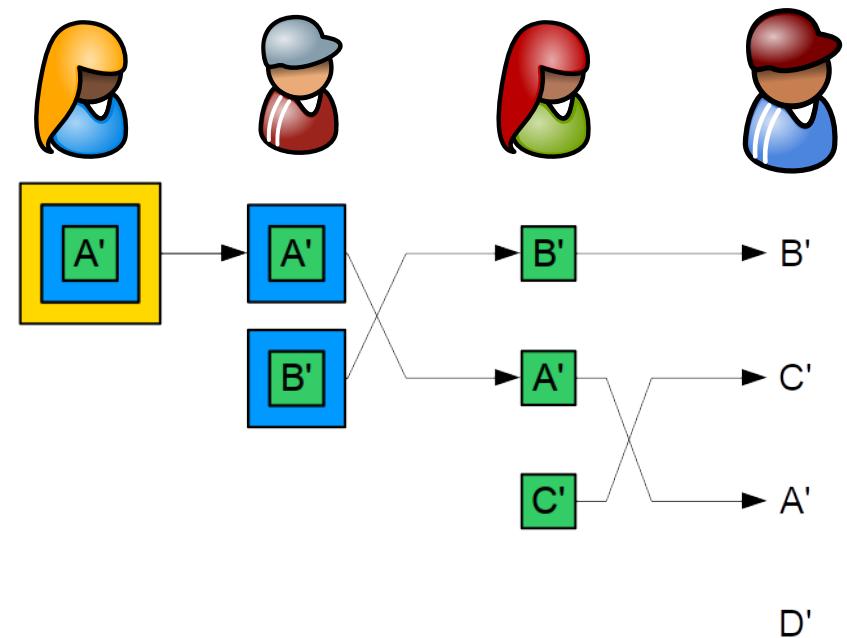
Traditional mixnet run by all peers

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

# State of the Art (I)

Traditional mixnet run by all peers

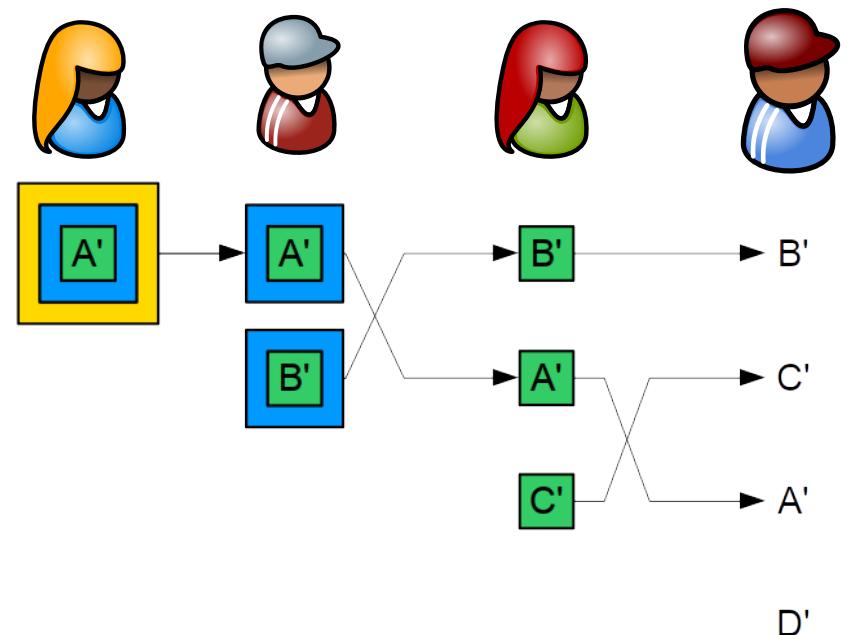
- Dissent (shuffle protocol) [CCS 2010],  
CoinShuffle [ESORICS 2014]



# State of the Art (I)

Traditional mixnet run by all peers

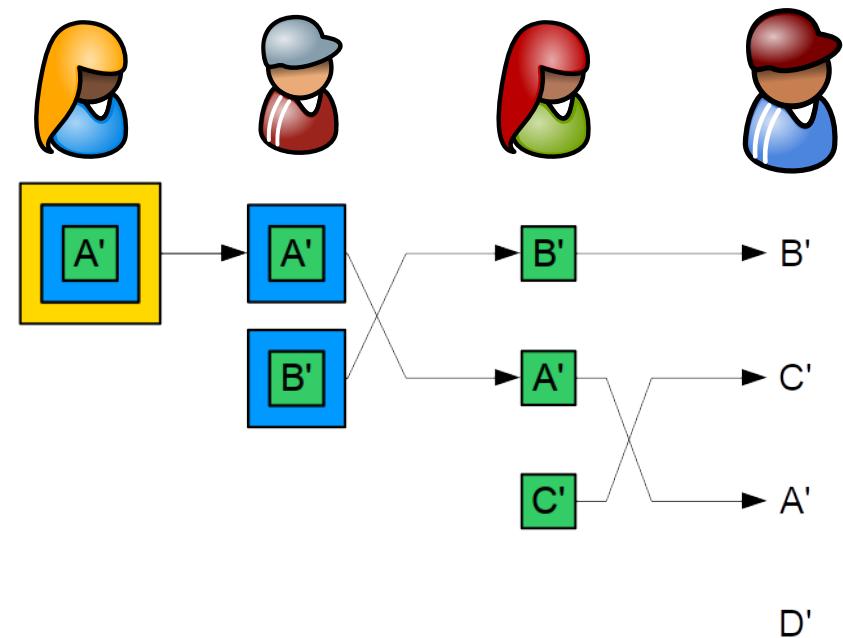
- Dissent (shuffle protocol) [CCS 2010], CoinShuffle [ESORICS 2014]
- $O(n)$  rounds in optimistic case



# State of the Art (I)

Traditional mixnet run by all peers

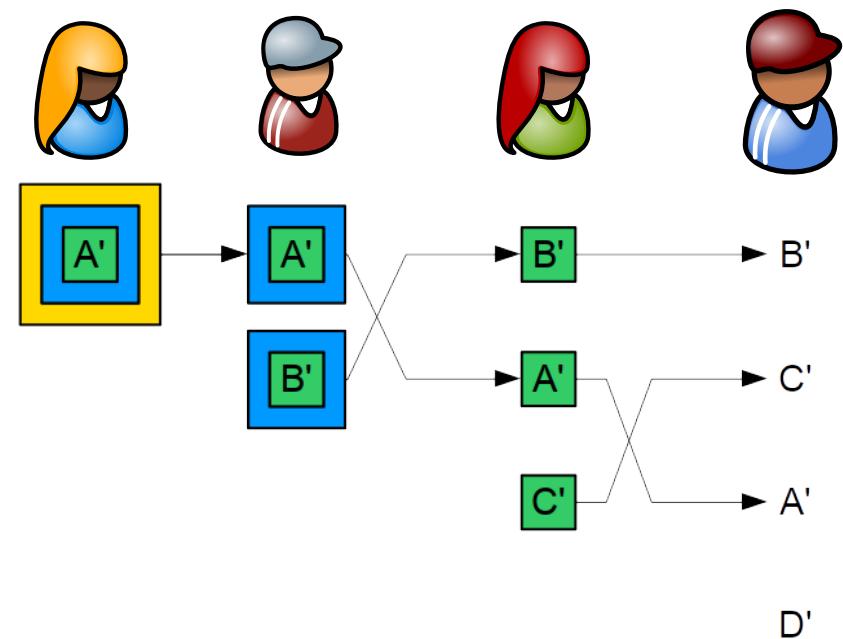
- Dissent (shuffle protocol) [CCS 2010], CoinShuffle [ESORICS 2014]
- $O(n)$  rounds in optimistic case
- $O(nf)$  rounds for  $f$  malicious peers



# 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!

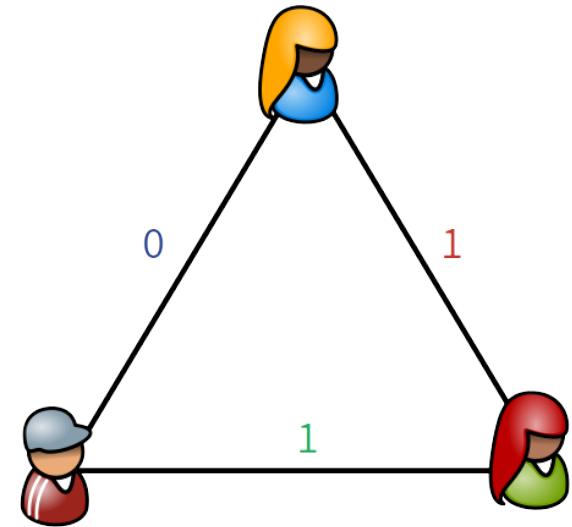
# **State of the Art (II)**

## **State of the Art (II)**

Dining cryptographers' networks (DC-nets)

## State of the Art (II)

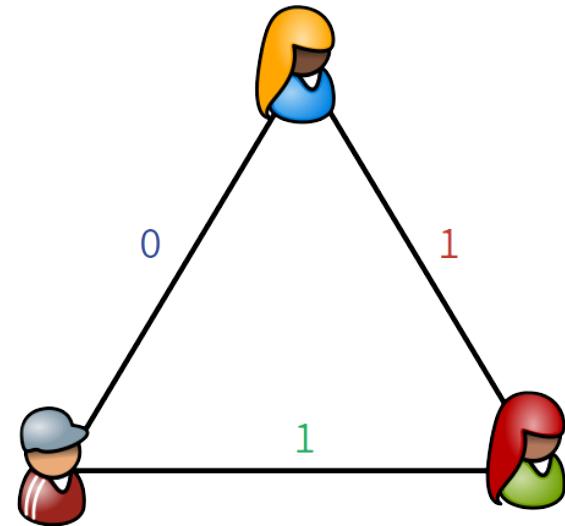
Dining cryptographers' networks (DC-nets)



## State of the Art (II)

Dining cryptographers' networks (DC-nets)

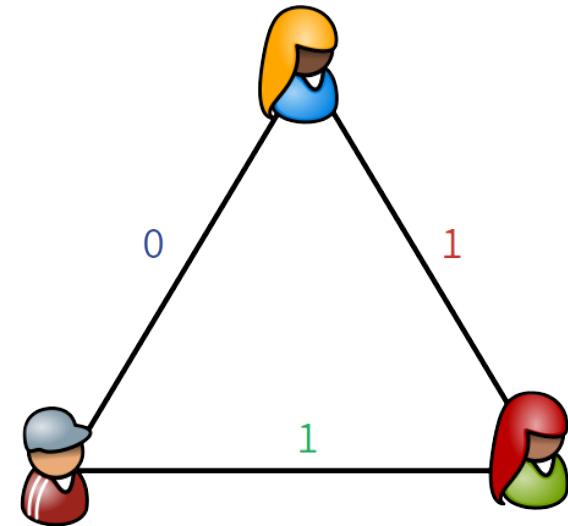
- Hope for  $O(1)$  rounds in the optimistic case



## State of the Art (II)

### Dining cryptographers' networks (DC-nets)

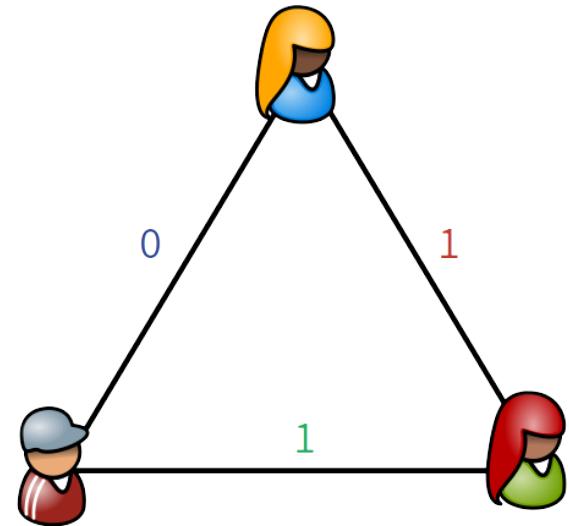
- Hope for  $O(1)$  rounds in the optimistic case
- Easy to disrupt



## State of the Art (II)

### Dining cryptographers' networks (DC-nets)

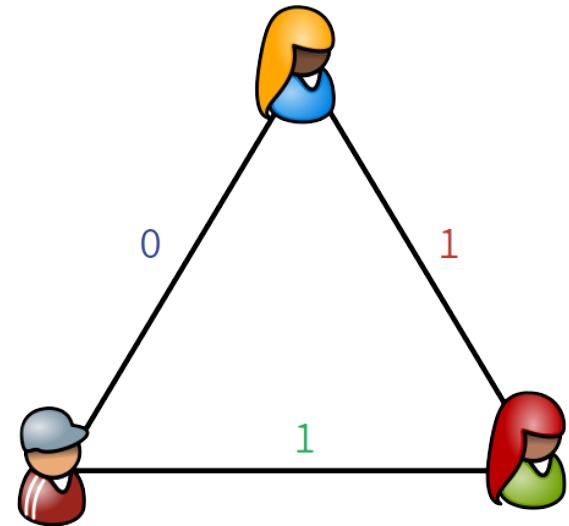
- Hope for  $O(1)$  rounds in the optimistic case
- Easy to disrupt
- All approaches to solve disruption problem suffer from drawbacks



## State of the Art (II)

### Dining cryptographers' networks (DC-nets)

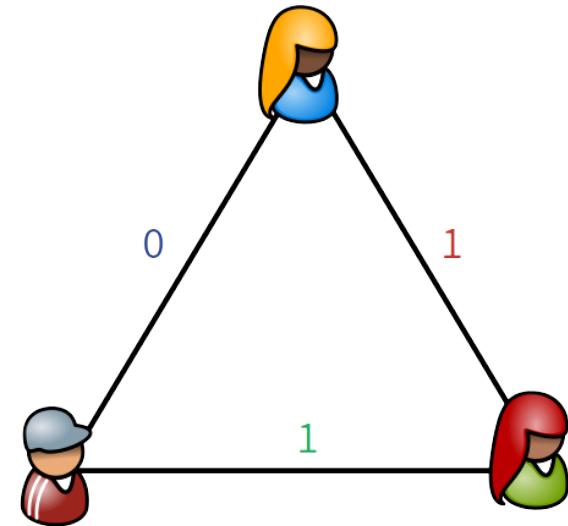
- Hope for  $O(1)$  rounds in the optimistic case
- Easy to disrupt
- All approaches to solve disruption problem suffer from drawbacks
- Golle and Juels [EUROCRYPT 2004]: Honest majority



## 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

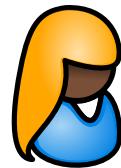


No practical P2P mixing protocol based on DC-nets!

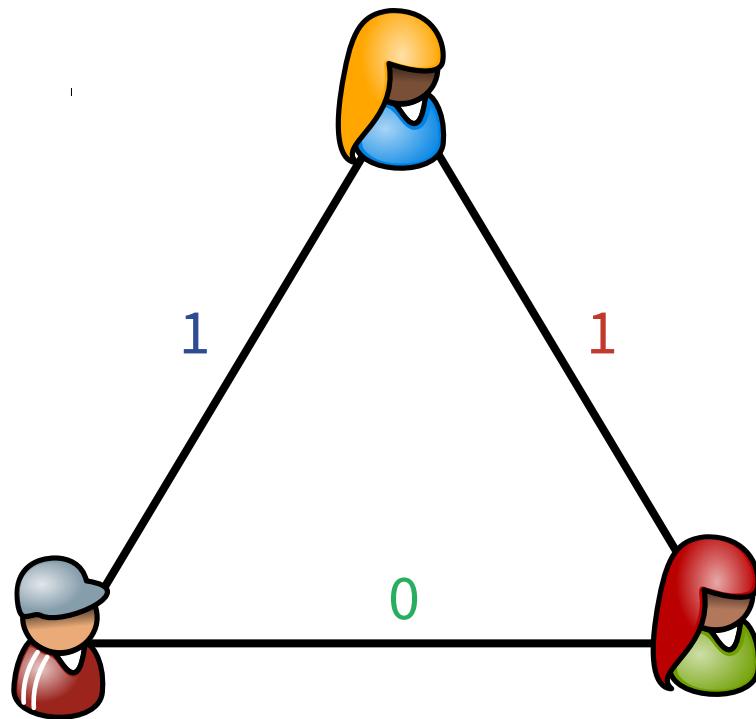
# DiceMix

A Practical P2P Mixing Protocol based on DC-nets

# DC-net [Chaum 1988]

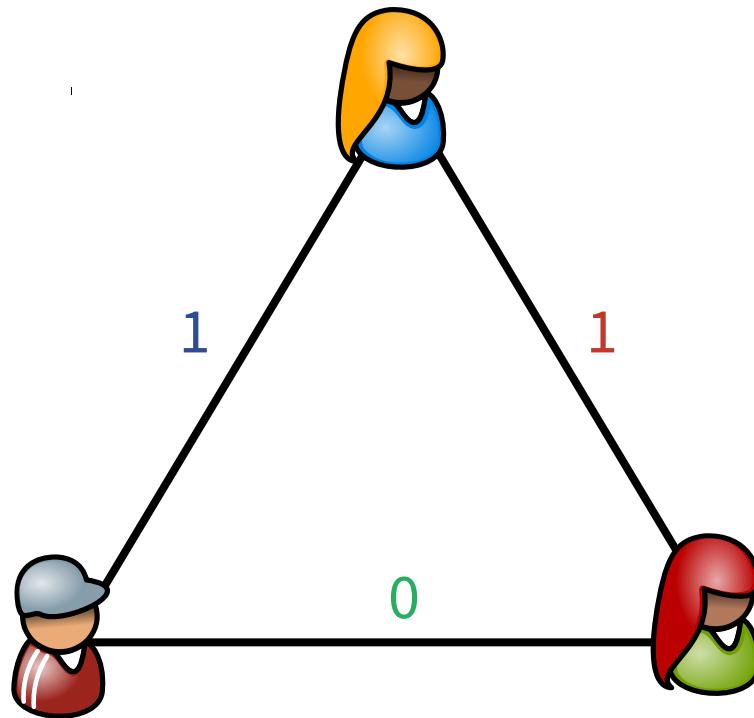


# DC-net [Chaum 1988]



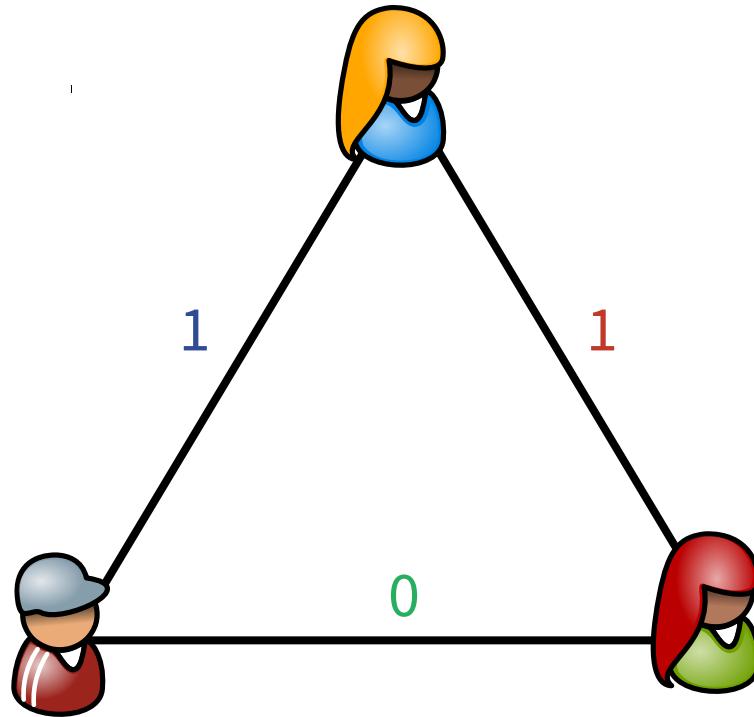
## DC-net [Chaum 1988]

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



## DC-net [Chaum 1988]

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

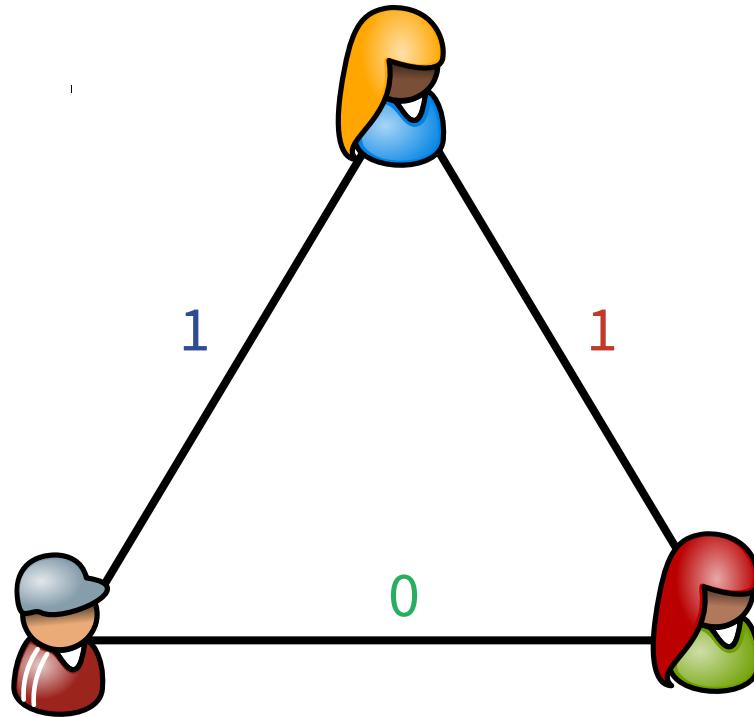


$$\mathbf{0} + (\mathbf{1} + \mathbf{0}) = \mathbf{1}$$

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

## DC-net [Chaum 1988]

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



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

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

$$1 + 1 + 0 = 0$$

# 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:	$m_1$	$m_1^2$	$m_1^3$	$\dots$	$m_1^n$
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 $n$ :	$m_n$	$m_n^2$	$m_n^3$	$\dots$	$m_n^n$
<hr/>					
	$\sum_{i=1}^n m_i$	$\sum_{i=1}^n m_i^2$	$\sum_{i=1}^n m_i^3$	$\dots$	$\sum_{i=1}^n m_i^n$

# 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 $n$ :	$m_n$	$m_n^2$	$m_n^3$	$\dots$	$m_n^n$
<hr/>					
	$\sum_{i=1}^n m_i$	$\sum_{i=1}^n m_i^2$	$\sum_{i=1}^n m_i^3$	$\dots$	$\sum_{i=1}^n m_i^n$

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

# Sending Several Messages [Bos, Boer 1989]

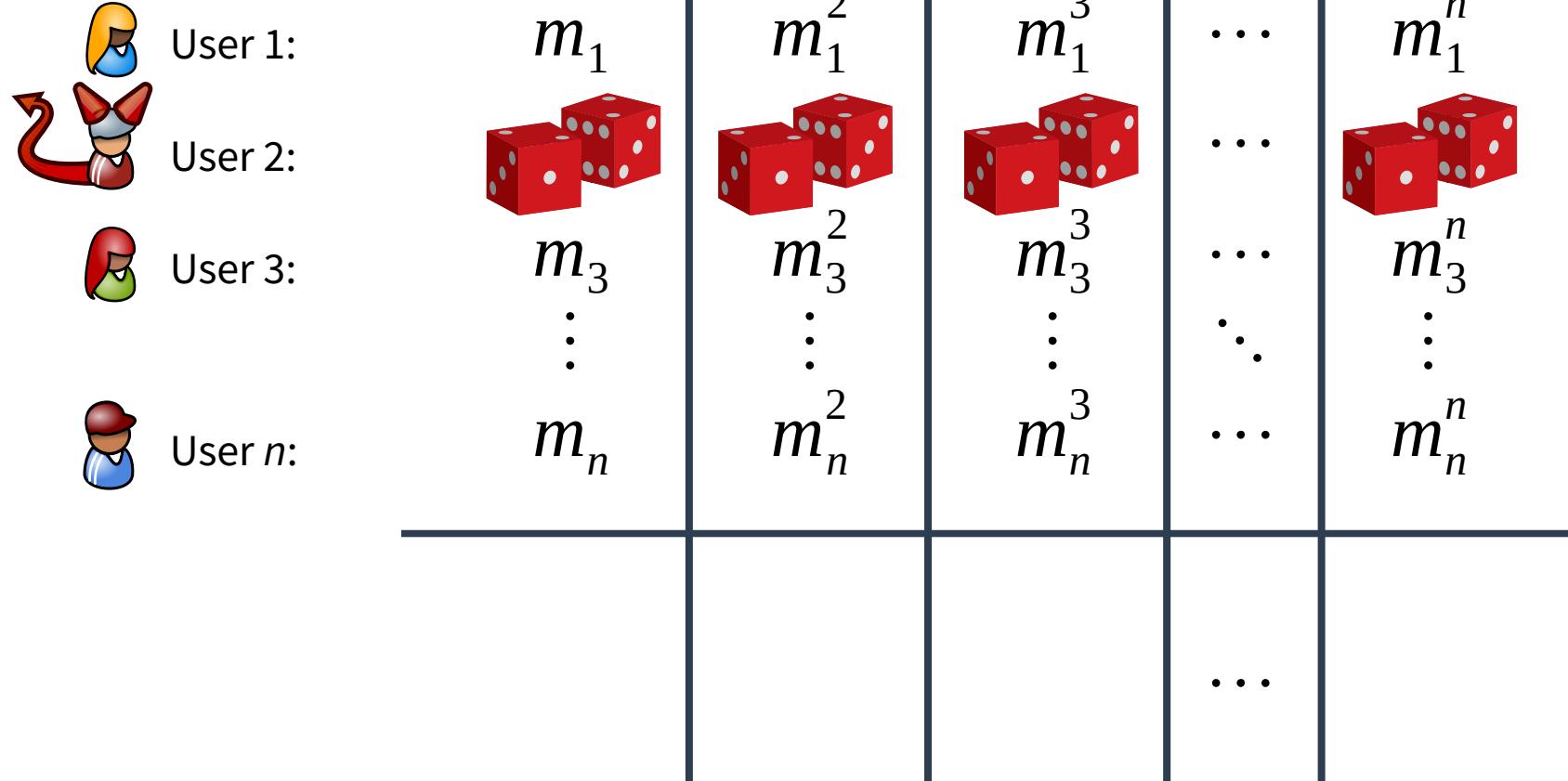
User 1:	$m_1$	$m_1^2$	$m_1^3$	...	$m_1^n$
User 2:	$m_2$	$m_2^2$	$m_2^3$	...	$m_2^n$
User 3:	$m_3$	$m_3^2$	$m_3^3$	...	$m_3^n$
	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
User $n$ :	$m_n$	$m_n^2$	$m_n^3$	...	$m_n^n$
<hr/>					
	$\sum_{i=1}^n m_i$	$\sum_{i=1}^n m_i^2$	$\sum_{i=1}^n m_i^3$	...	$\sum_{i=1}^n m_i^n$

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

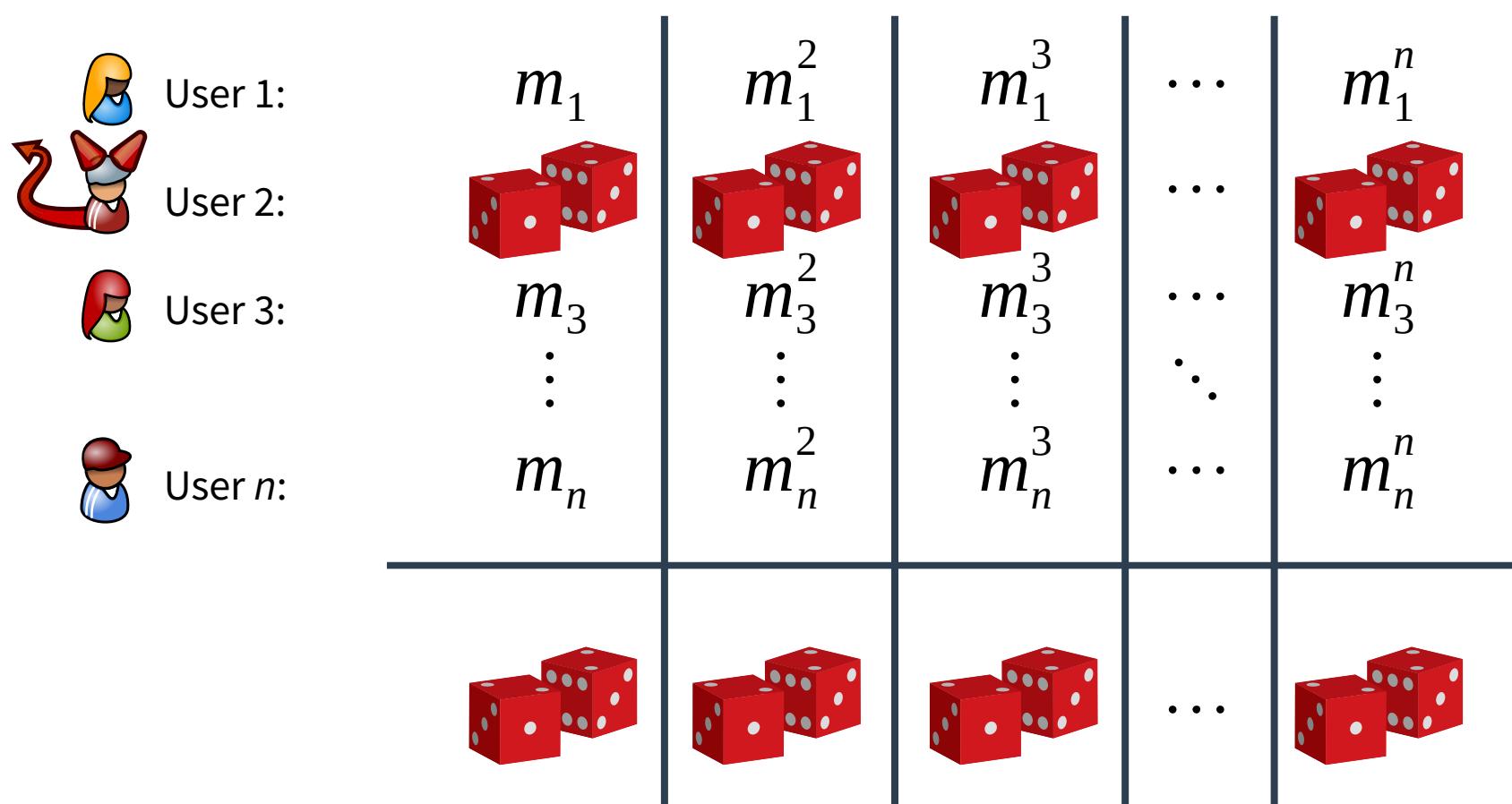
# Disruption

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

# Disruption



# Disruption



Malicious user stays anonymous!

# Handling Disruptions

# Handling Disruptions

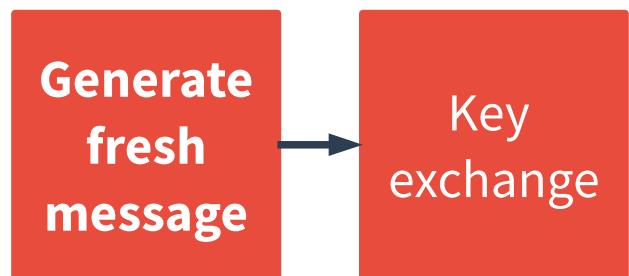
**IN CASE OF  
DISRUPTION  
BREAK ANONYMITY**



# Flowchart of DiceMix

Generate  
fresh  
message

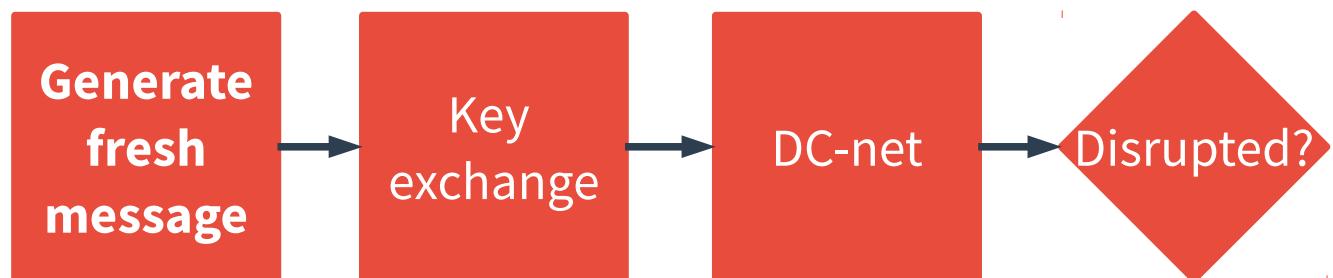
# Flowchart of DiceMix



# Flowchart of DiceMix



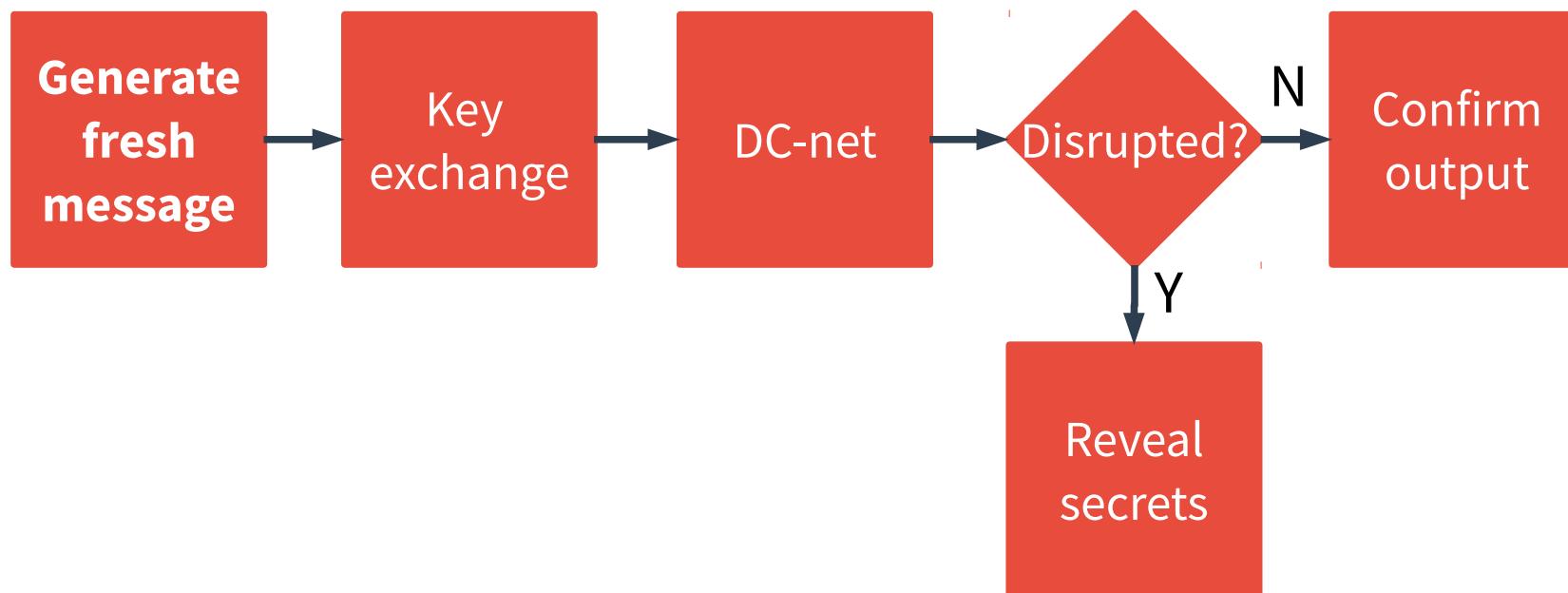
# Flowchart of DiceMix



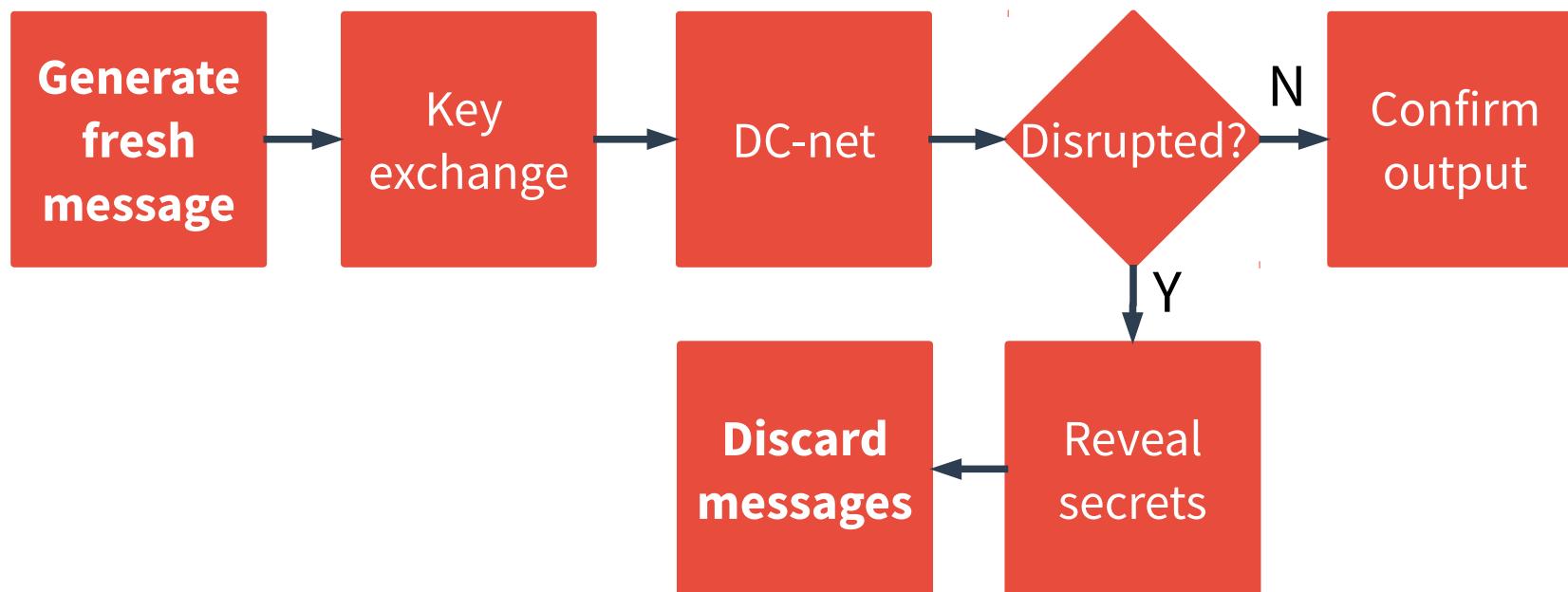
# Flowchart of DiceMix



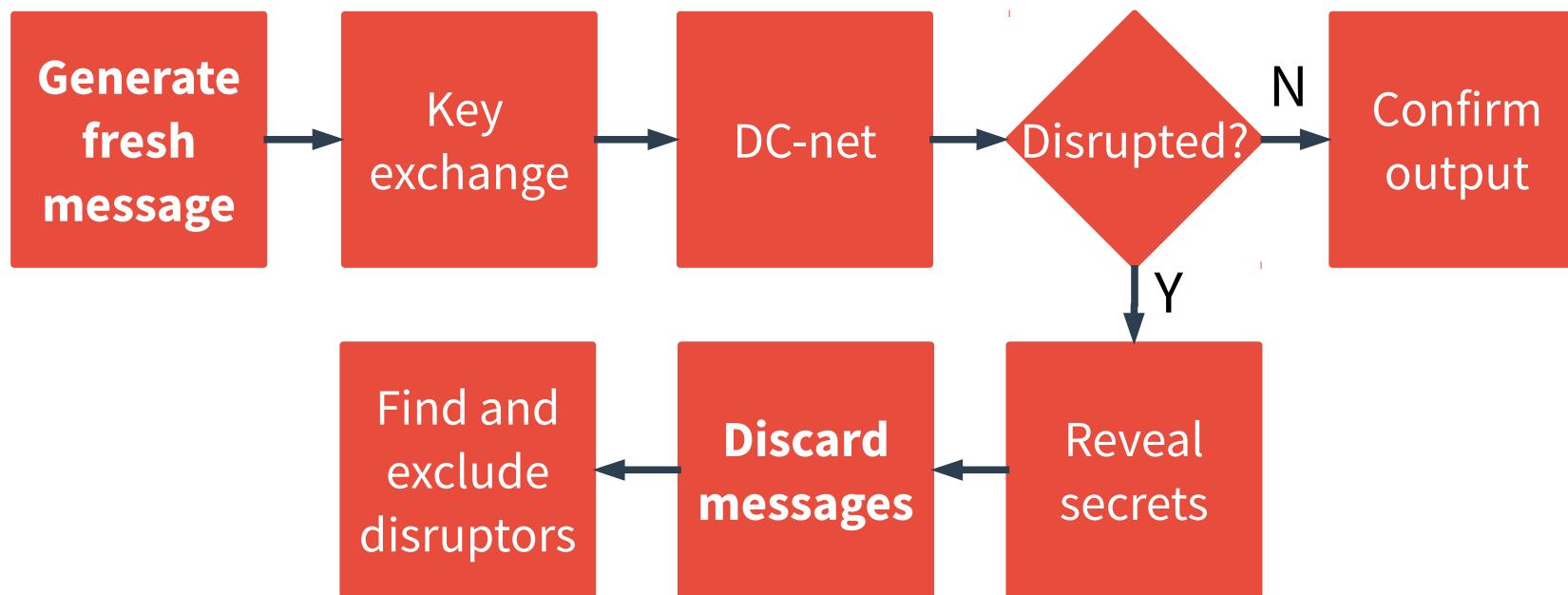
# Flowchart of DiceMix



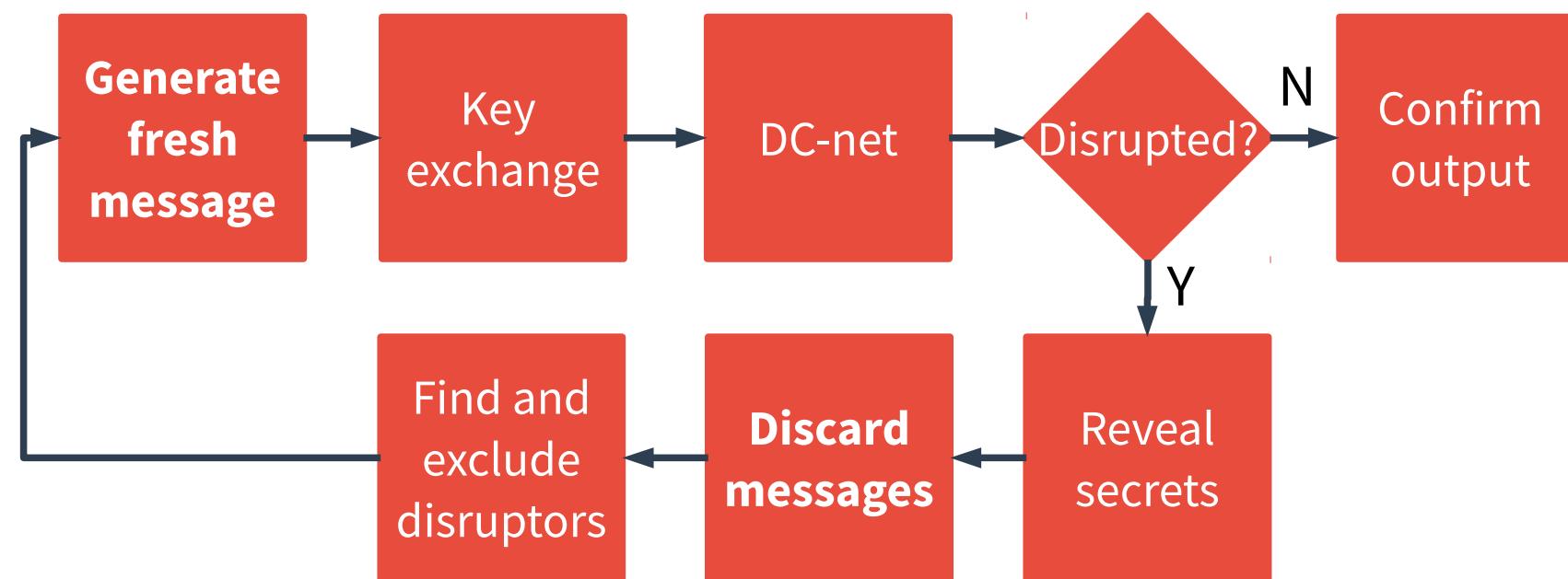
# Flowchart of DiceMix



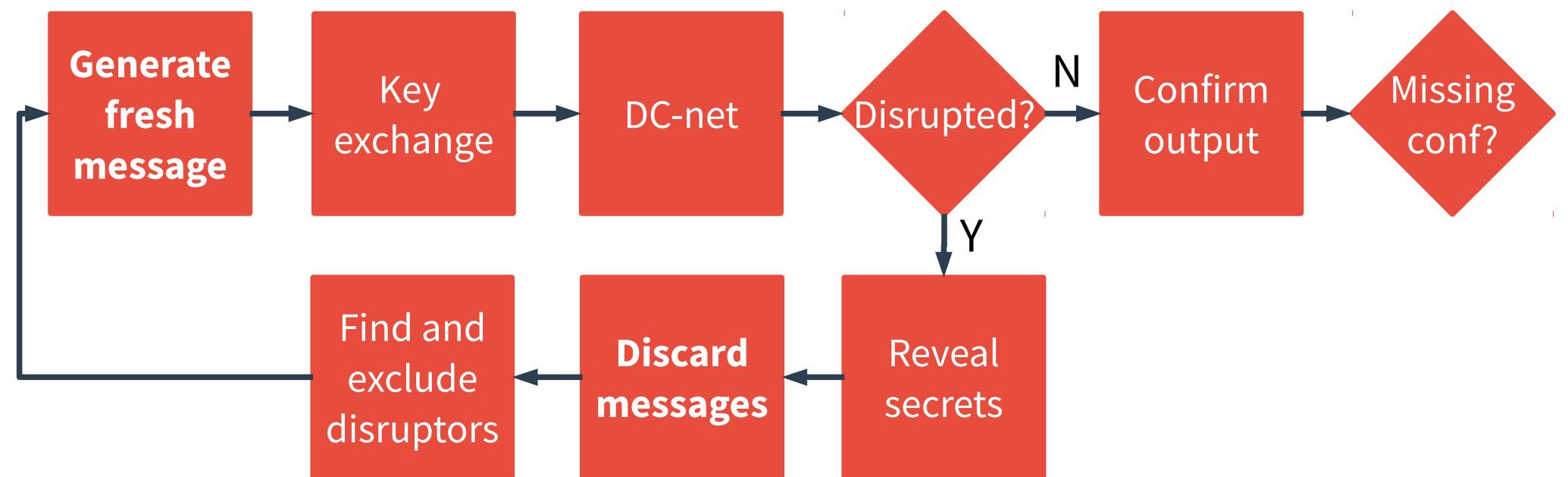
# Flowchart of DiceMix



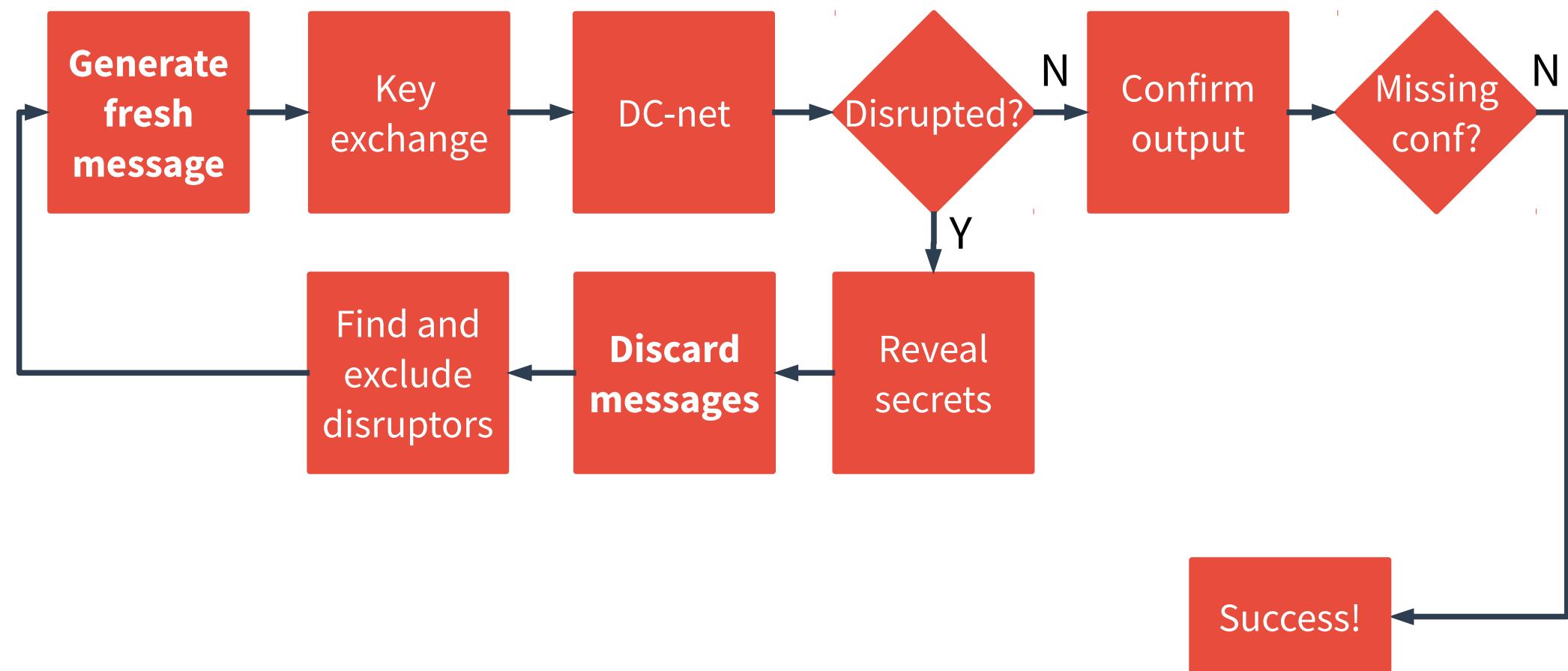
# Flowchart of DiceMix



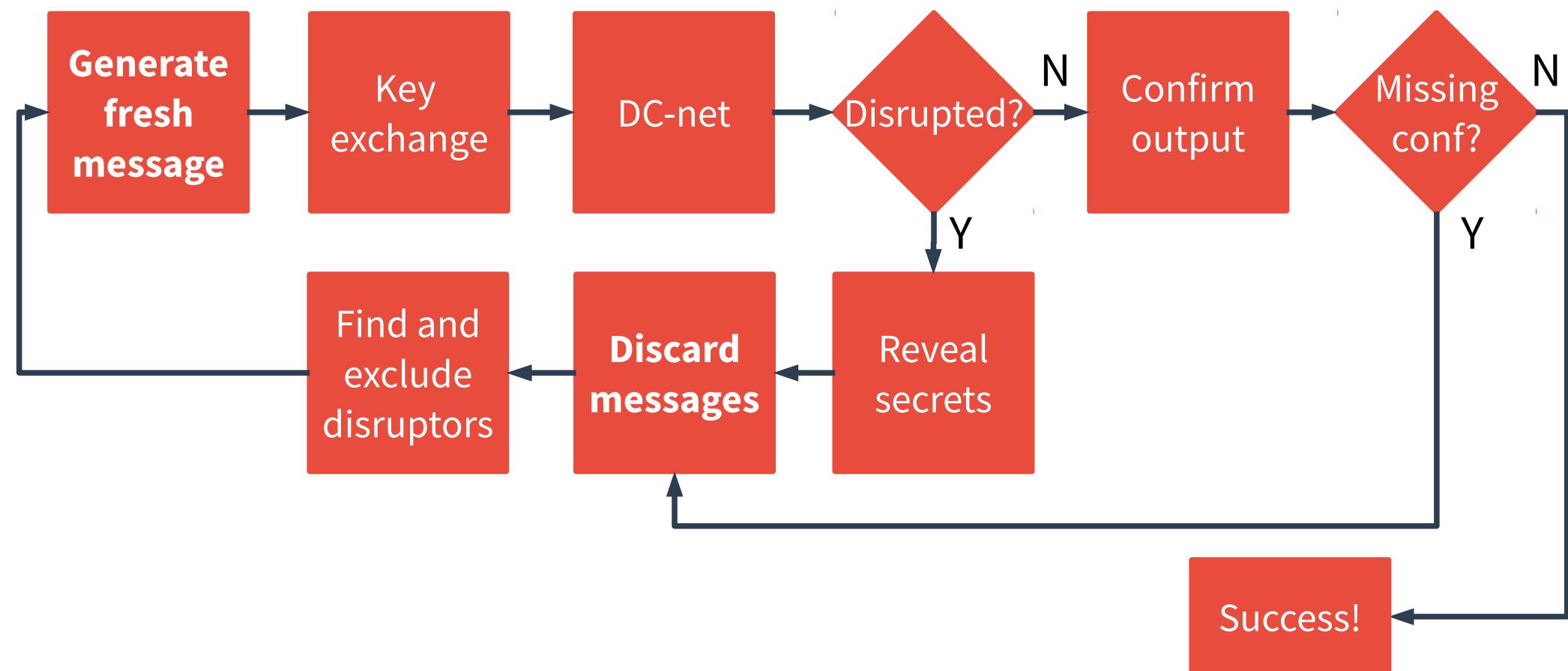
# Flowchart of DiceMix



# Flowchart of DiceMix

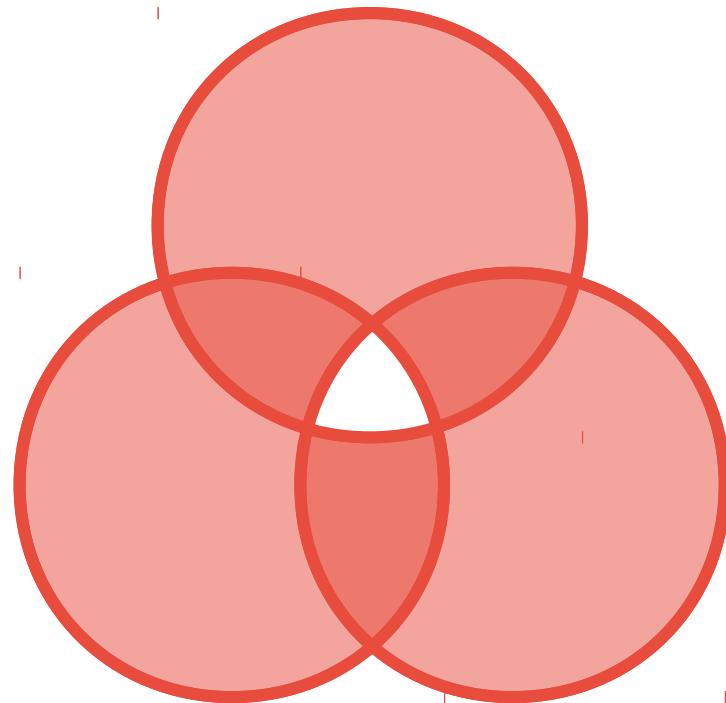


# Flowchart of DiceMix



# Freshness Is Necessary

Anonymity



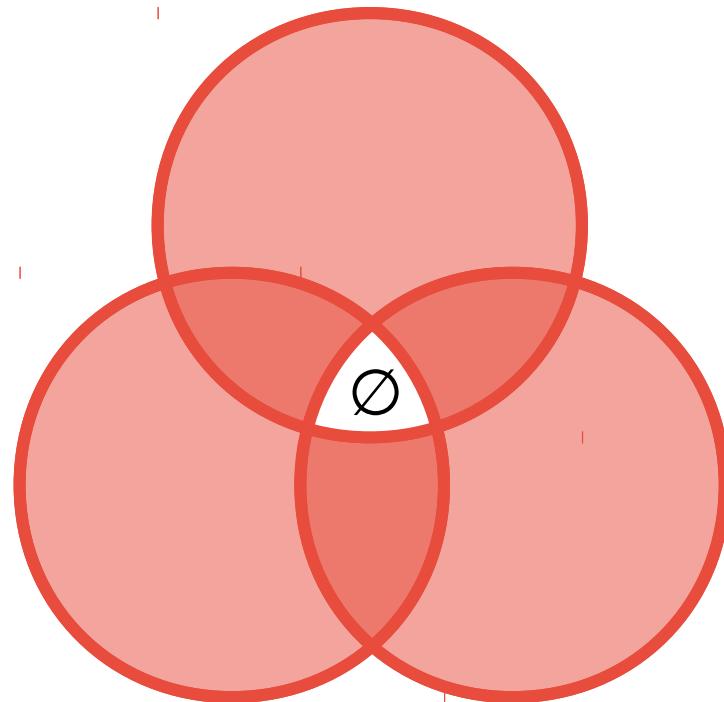
Termination with  
dishonest majority

Support for  
fixed messages



# Freshness Is Necessary

Anonymity

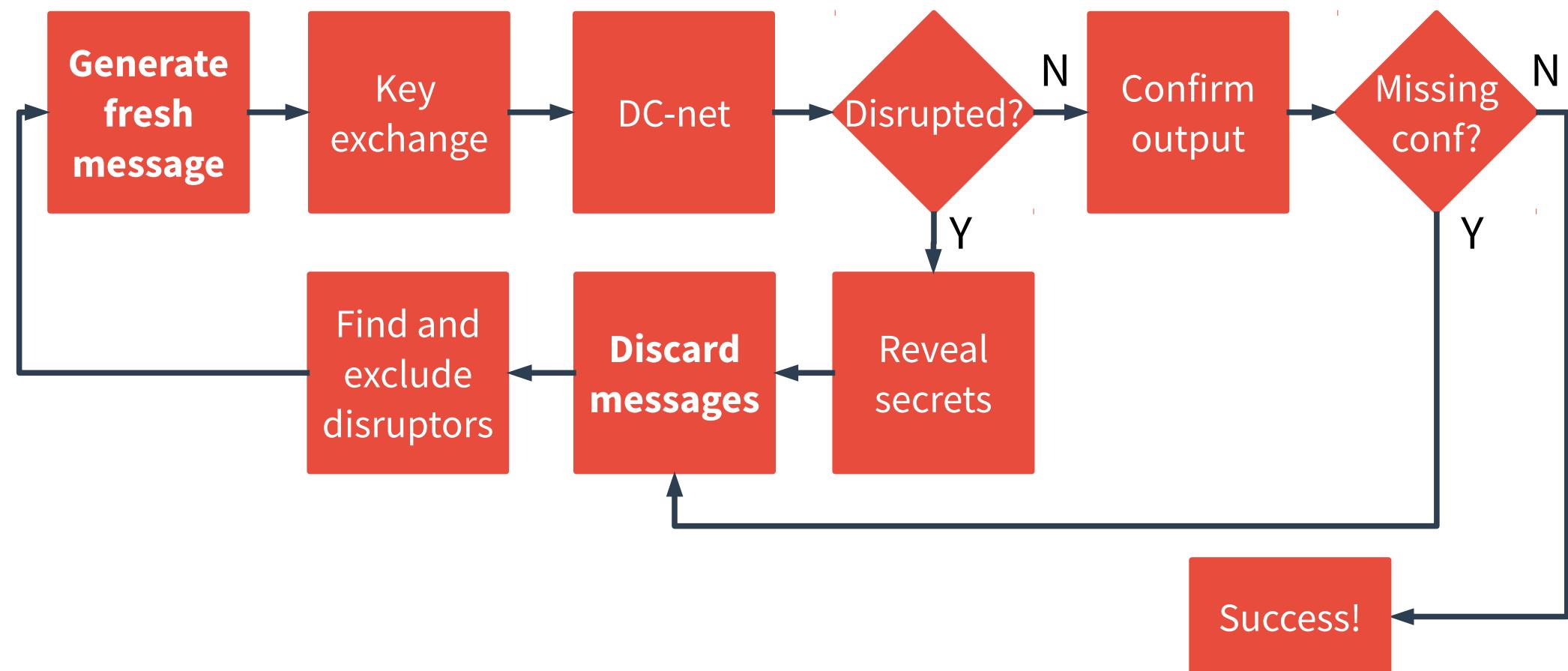


Termination with  
dishonest majority

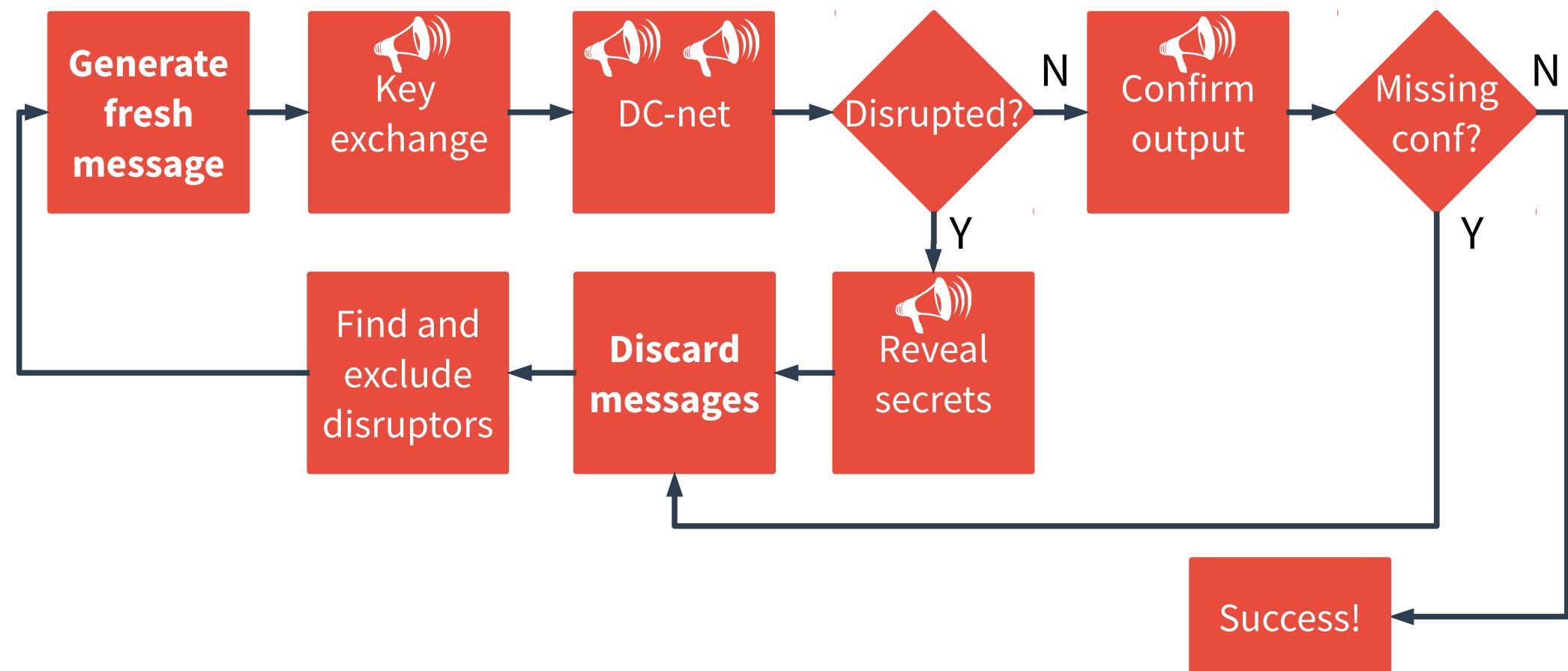
Support for  
fixed messages



# Flowchart of DiceMix

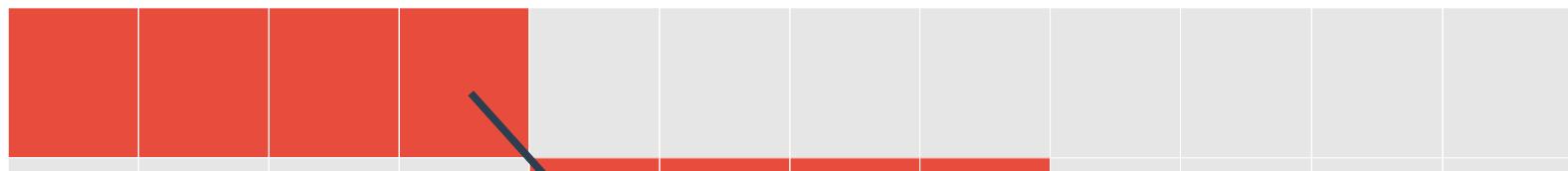


# Flowchart of DiceMix

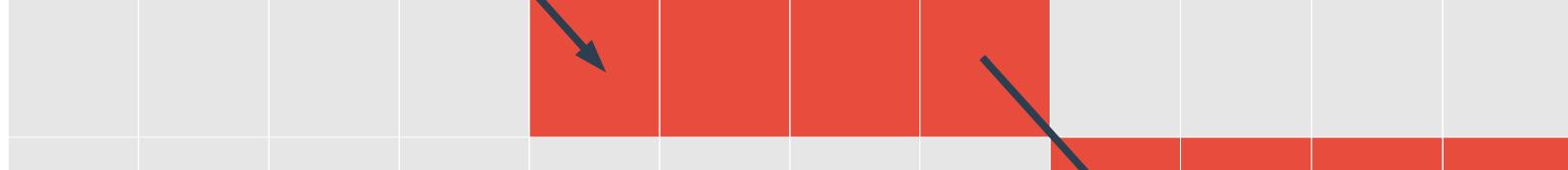


# Communication Rounds (naive)

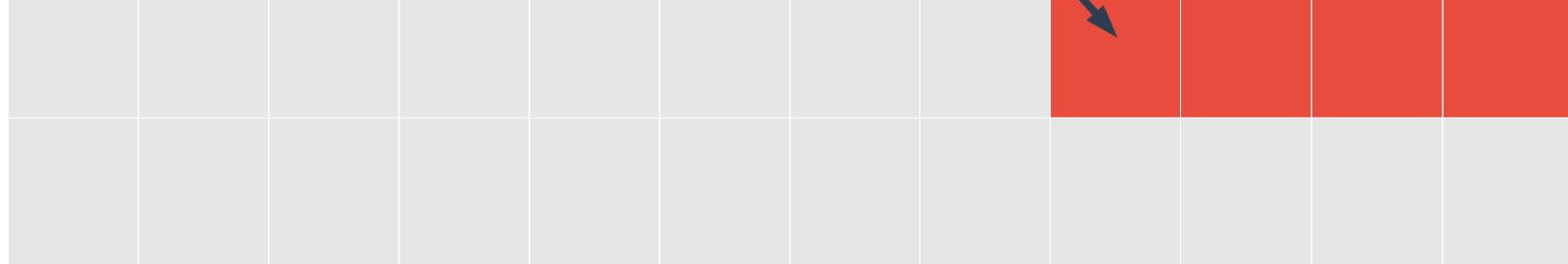
Run 1



Run 2

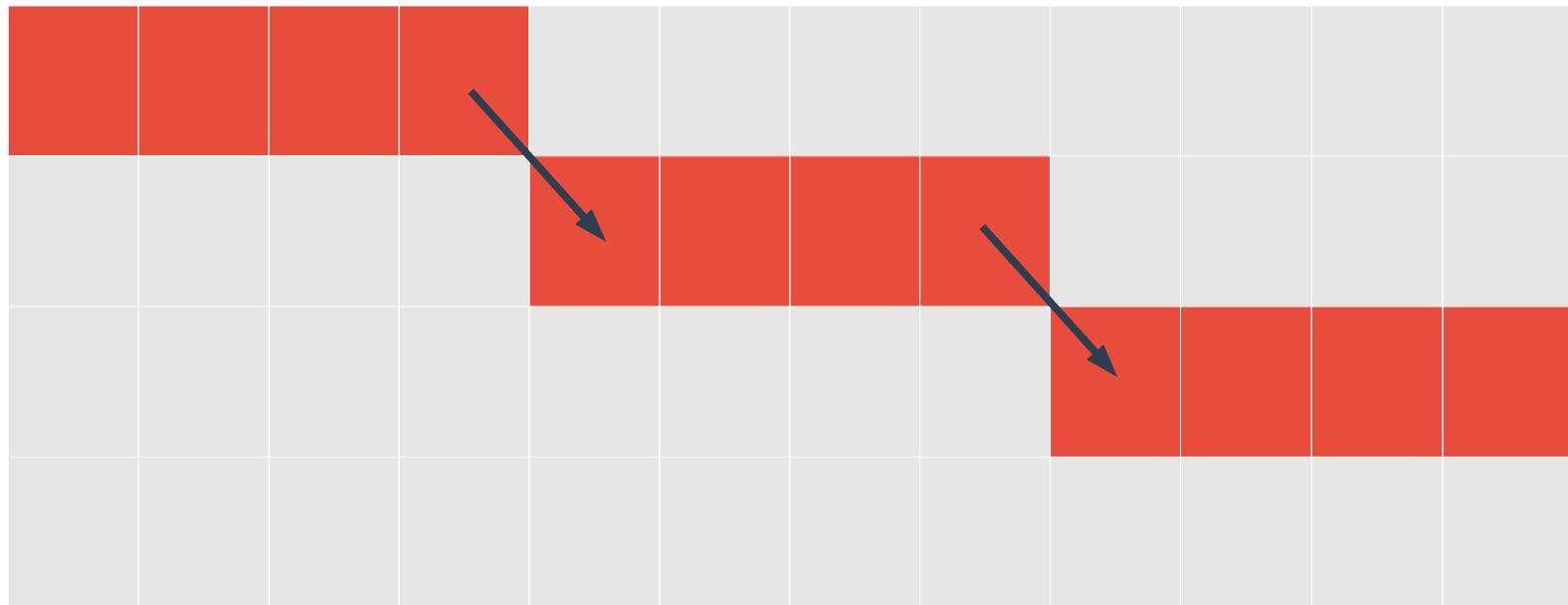


Run 3



# Communication Rounds (naive)

Run 1



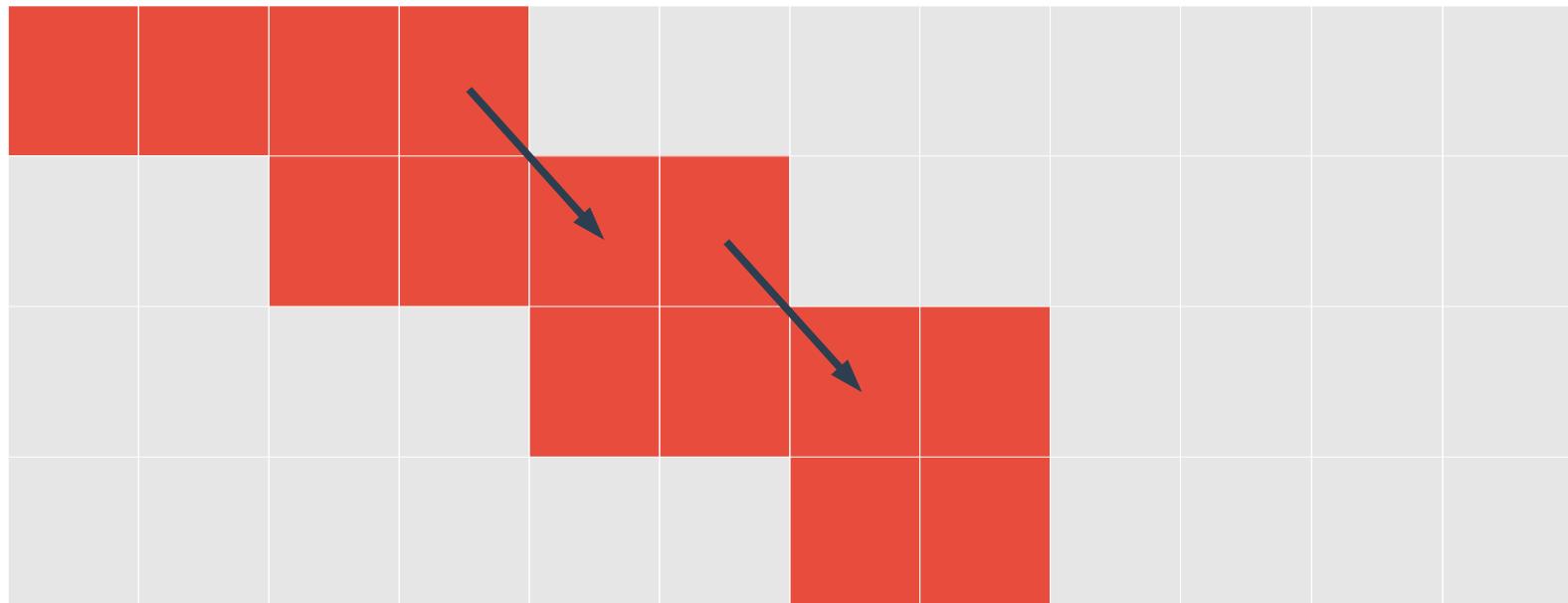
Run 2

Run 3

$4 + 4f$  rounds

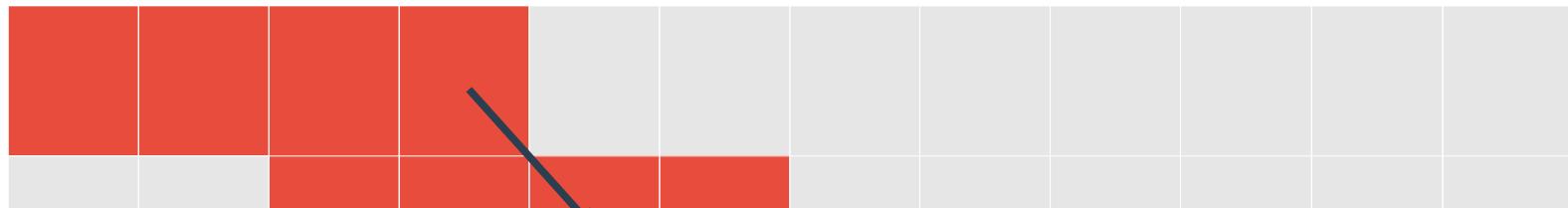
# Communication Rounds (DiceMix)

Run 1  
Run 2  
Run 3  
(Run 4)

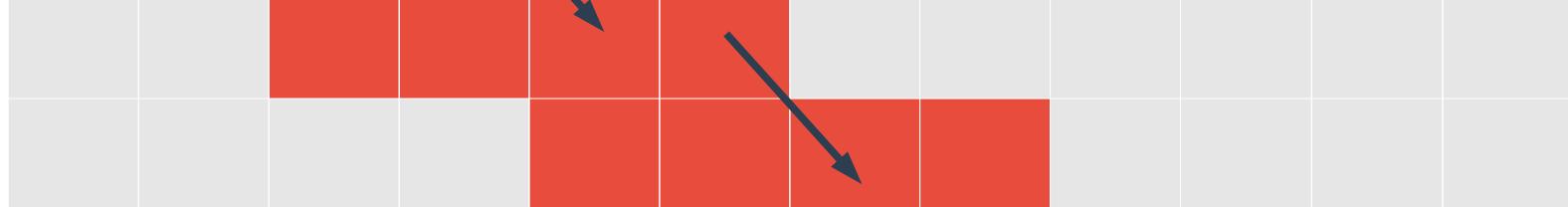


# Communication Rounds (DiceMix)

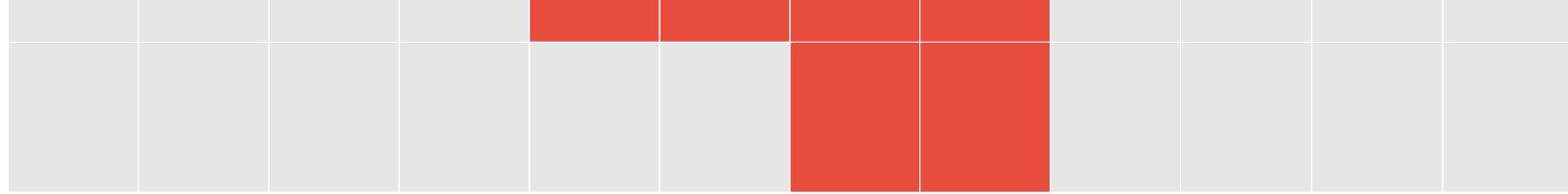
Run 1



Run 2



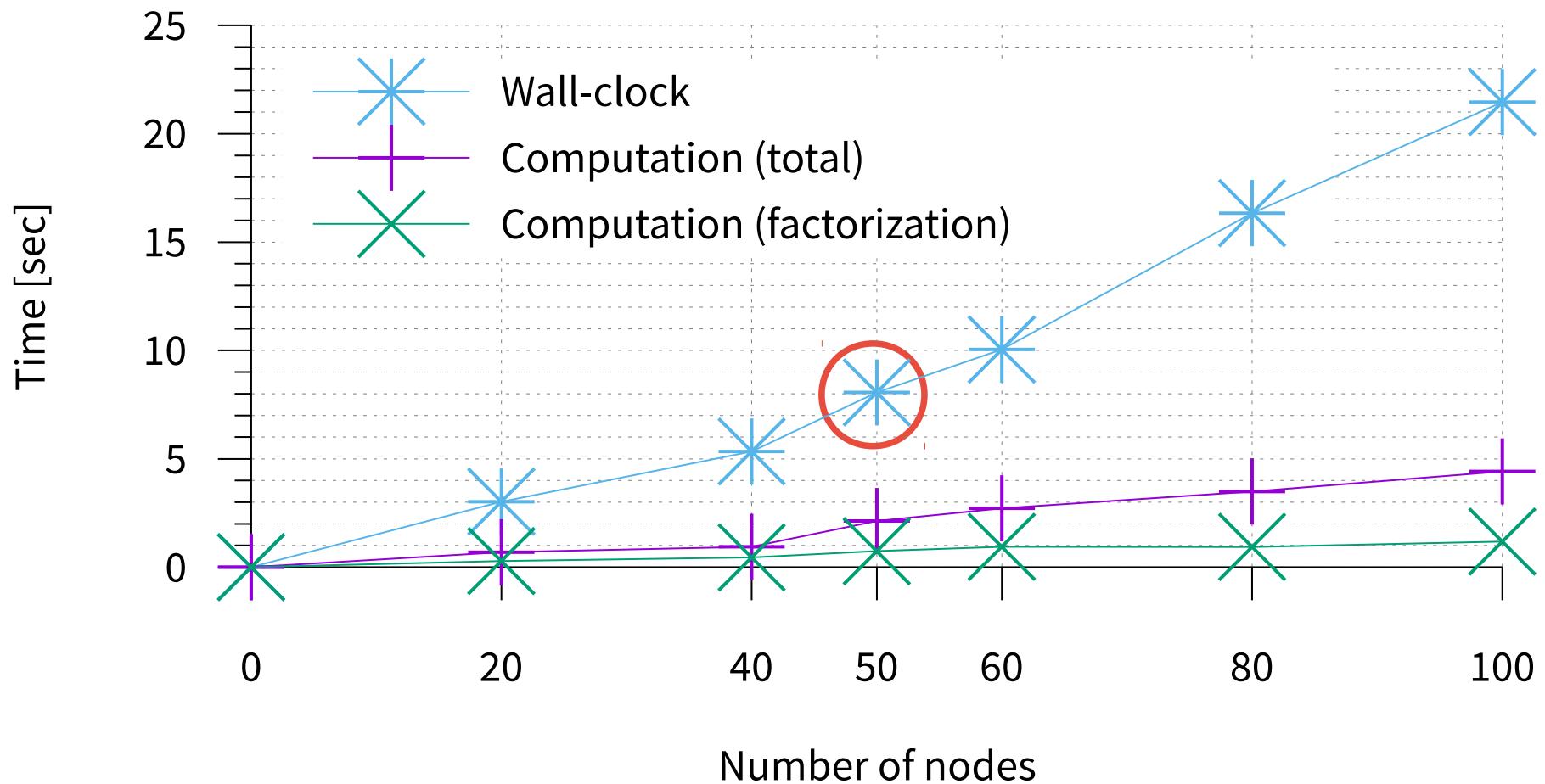
Run 3



(Run 4)

$4 + 2f$  rounds

# Performance



# **CoinShuffle++**

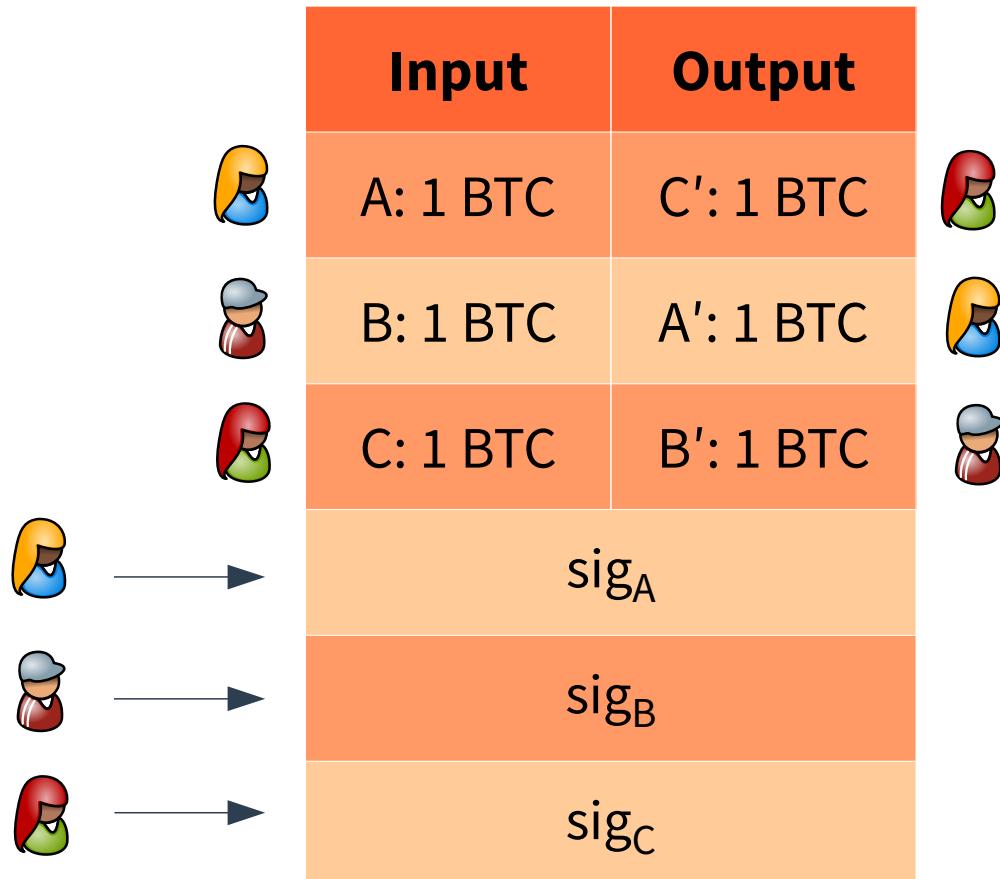
**A P2P Coin Mixing Protocol based on DiceMix**

# 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	

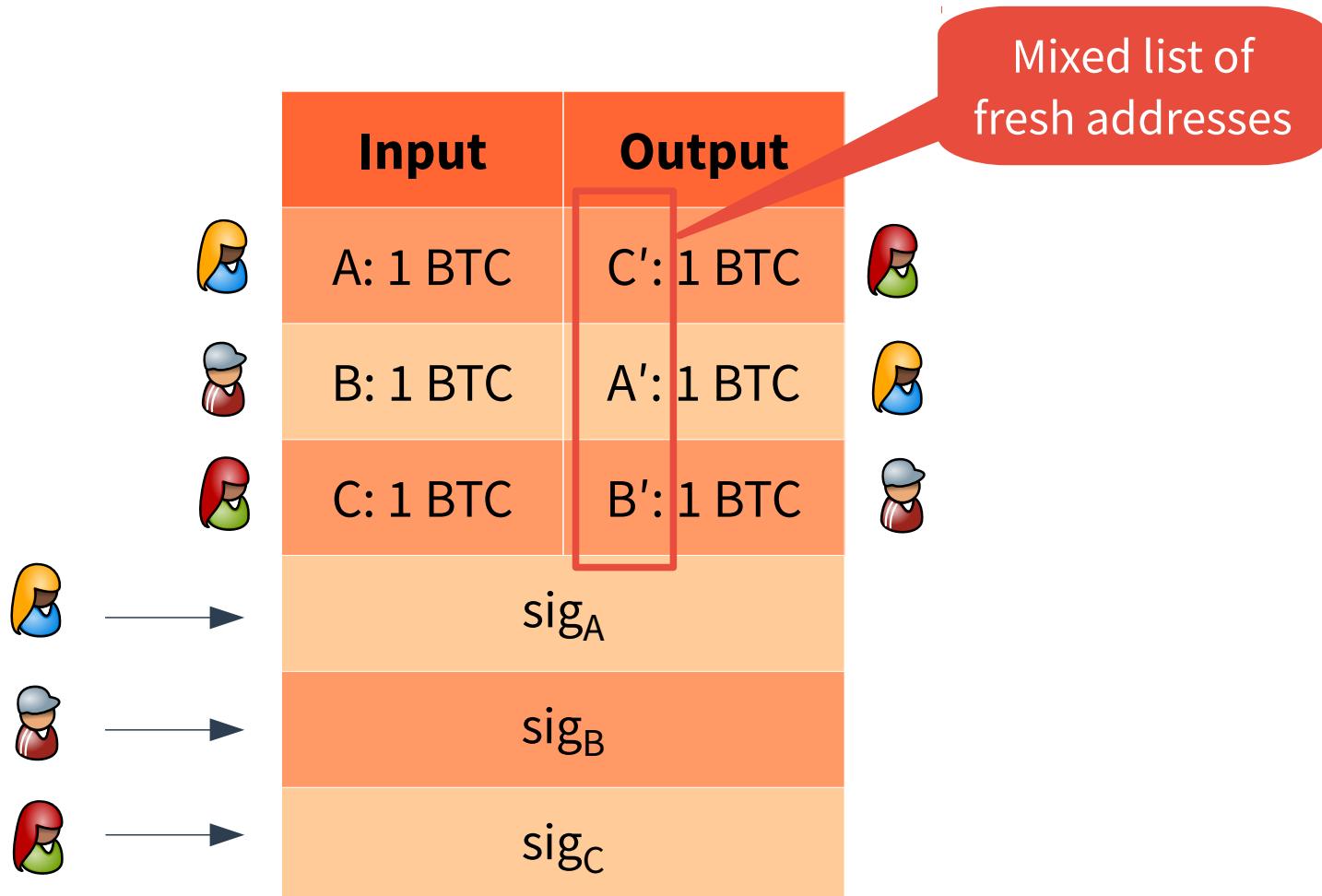
[CoinJoin, Maxwell 2013]

# Mixing without a Third Party (CoinJoin)



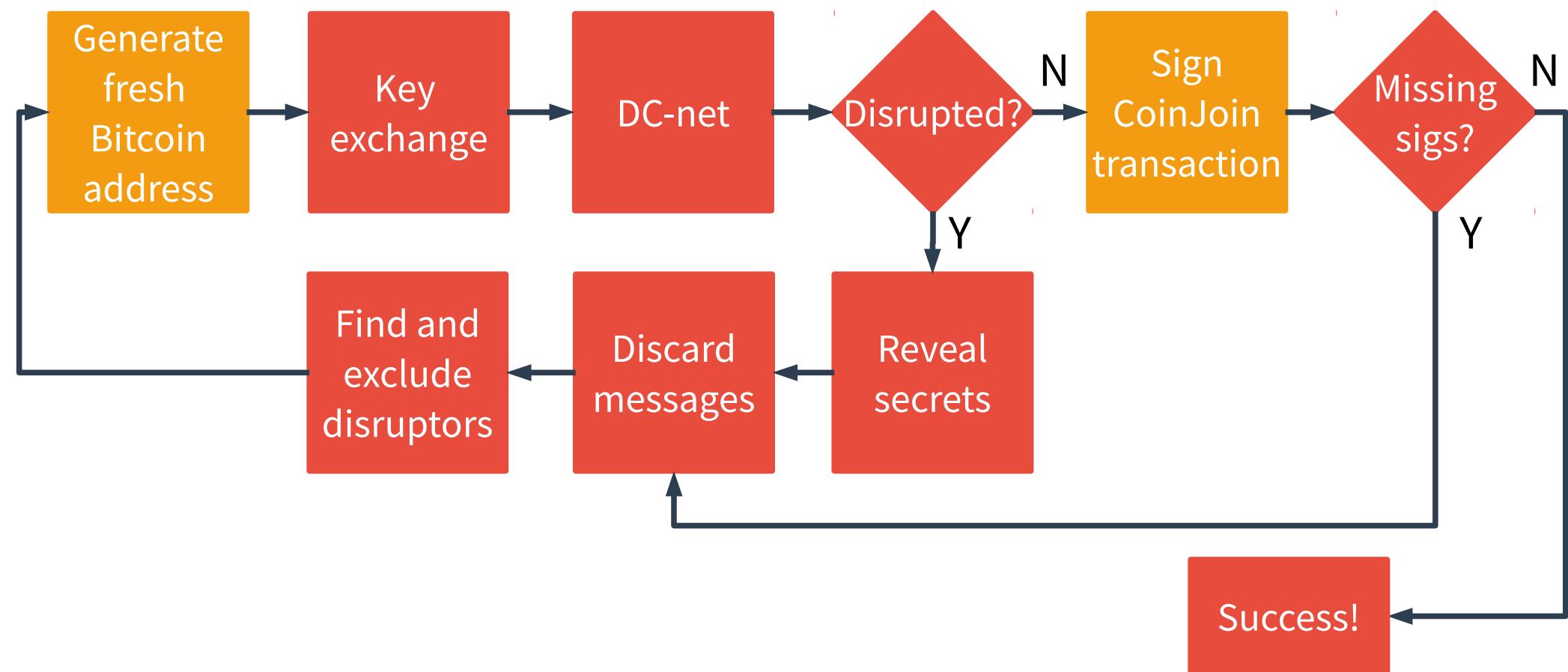
[CoinJoin, Maxwell 2013]

# Mixing without a Third Party (CoinJoin)



[CoinJoin, Maxwell 2013]

# Flowchart of CoinShuffle++



# Comparison of CoinShuffle++ vs. TumbleBit

	TumbleBit (classic tumbler)	CoinShuffle++
Anonymity set / payment	<b>&gt;&gt; 100</b>	~ 100
Bandwidth / payment	<b>~ 420 bytes</b>	~ 2250 bytes
Total running time / payment	<b>&lt; 5 s</b>	< 20 s
Coordination required	<b>no</b>	yes

} off-chain

# Comparison of CoinShuffle++ vs. TumbleBit

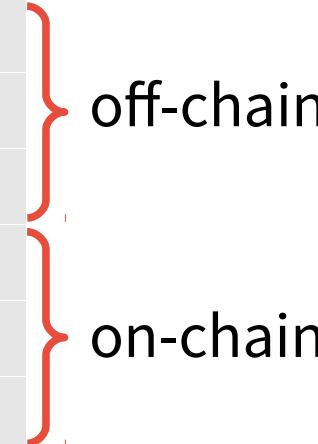
	TumbleBit (classic tumbler)	CoinShuffle++
Anonymity set / payment	<b>&gt;&gt; 100</b>	~ 100
Bandwidth / payment	<b>~ 420 bytes</b>	~ 2250 bytes
Total running time / payment	<b>&lt; 5 s</b>	< 20 s
Coordination required	<b>no</b>	yes
Sequential blocks / payment	<b>2 + 1</b>	<b>1 + 1</b>
Input-output pairs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

off-chain

on-chain

# Comparison of CoinShuffle++ vs. TumbleBit

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



The table compares the performance and features of TumbleBit (classic tumbler) and CoinShuffle++. The comparison includes metrics like anonymity set size, bandwidth usage, total running time, coordination required, sequential blocks, input-output pairs, centralization, collateral requirements, DoS/Sybil protection, and support for confidential transactions. A red curly brace on the right side of the table groups the 'off-chain' and 'on-chain' columns.

## **More Shuffling**

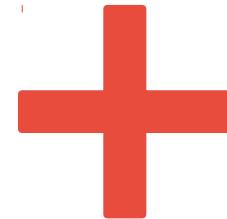
### **Shameless Plugs**

# ValueShuffle

ValueShuffle



CoinShuffle++



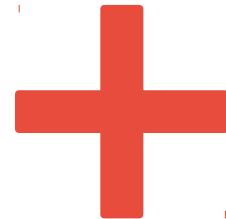
Confidential  
Transactions

# ValueShuffle

ValueShuffle



CoinShuffle++



Confidential  
Transactions

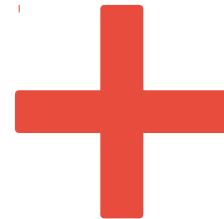
- Hides amounts in transactions and provides anonymity

# ValueShuffle

ValueShuffle



CoinShuffle++



Confidential  
Transactions

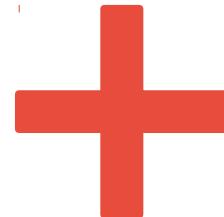
- Hides amounts in transactions and provides anonymity
- **Users with different amounts of money can mix!**

# ValueShuffle

ValueShuffle



CoinShuffle++



Confidential  
Transactions

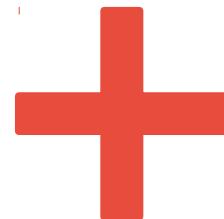
- Hides amounts in transactions and provides anonymity
- **Users with different amounts of money can mix!**
- User can mix and pay simultaneously in one transaction

# ValueShuffle

ValueShuffle



CoinShuffle++



Confidential  
Transactions

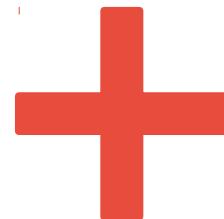
- Hides amounts in transactions and provides anonymity
- **Users with different amounts of money can mix!**
- User can mix and pay simultaneously in one transaction
- Accepted at Bitcoin Workshop 2017

# 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

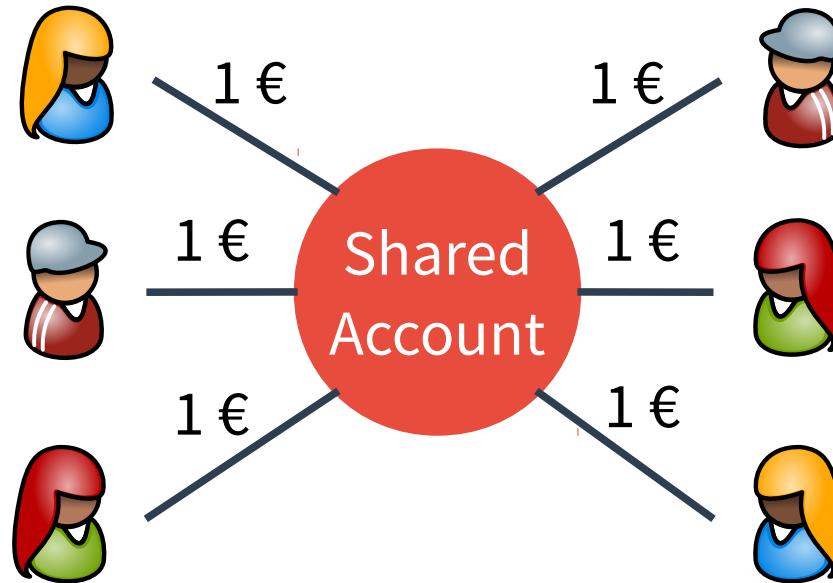
## Money Mixing in Credit Networks

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

# PathShuffle

## Money Mixing in Credit Networks

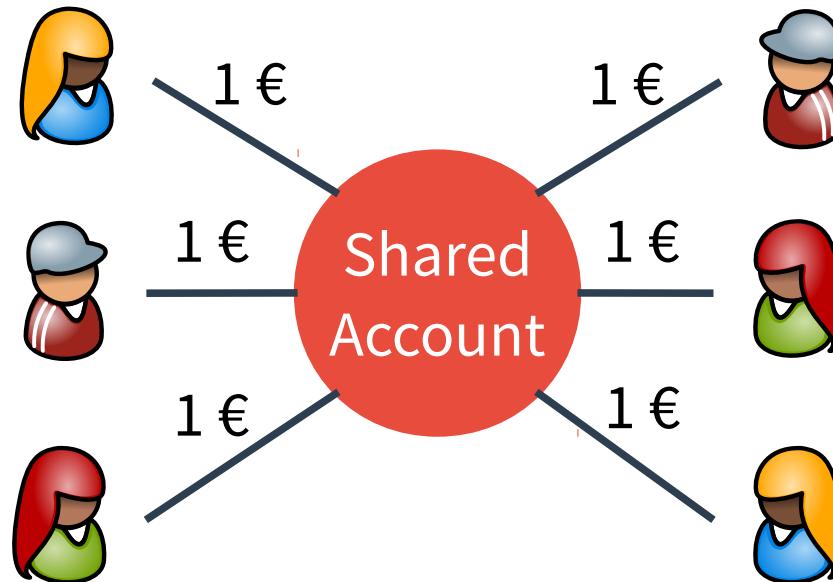
- Idea similar to CoinJoin, but there is no CoinJoin transaction
- Challenge: Simulate the CoinJoin via a shared account



# PathShuffle

## Money Mixing in Credit Networks

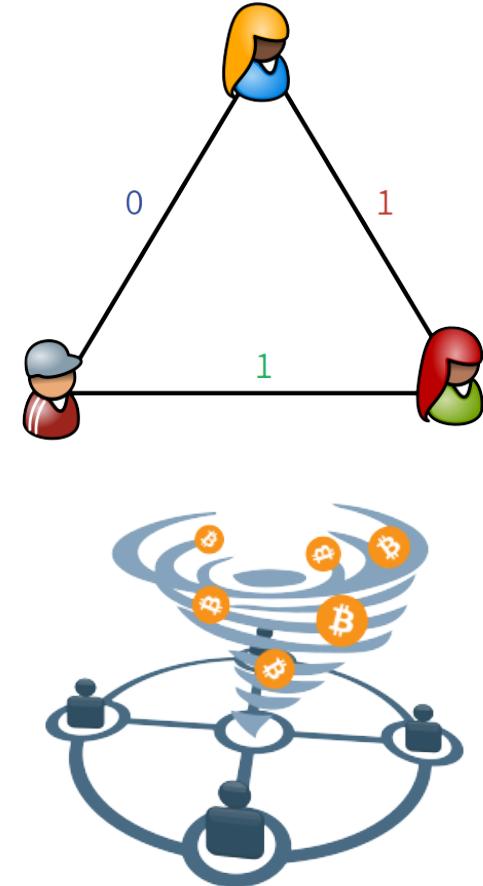
- Idea similar to CoinJoin, but there is no CoinJoin transaction
- Challenge: Simulate the CoinJoin via a shared account



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



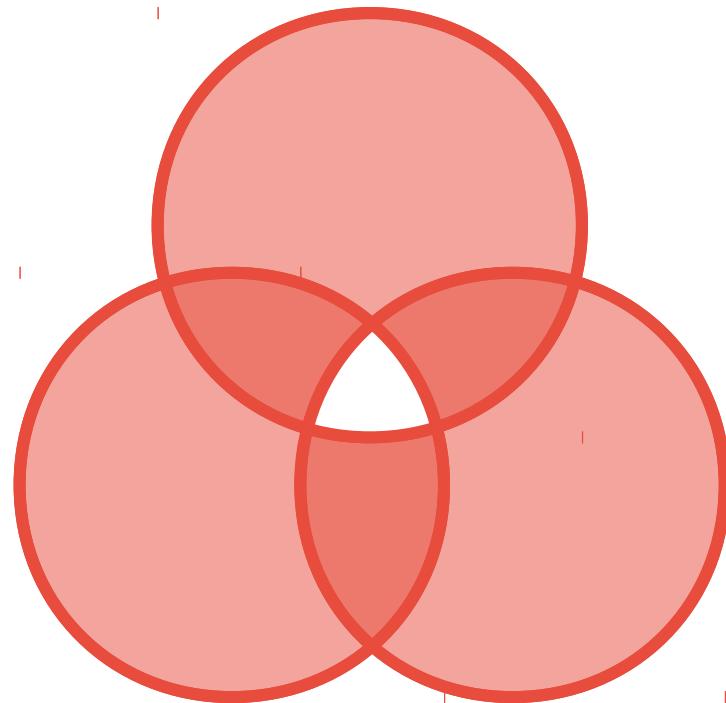
Work in progress:

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

# **Backup Slides**

# Freshness Is Necessary

Anonymity

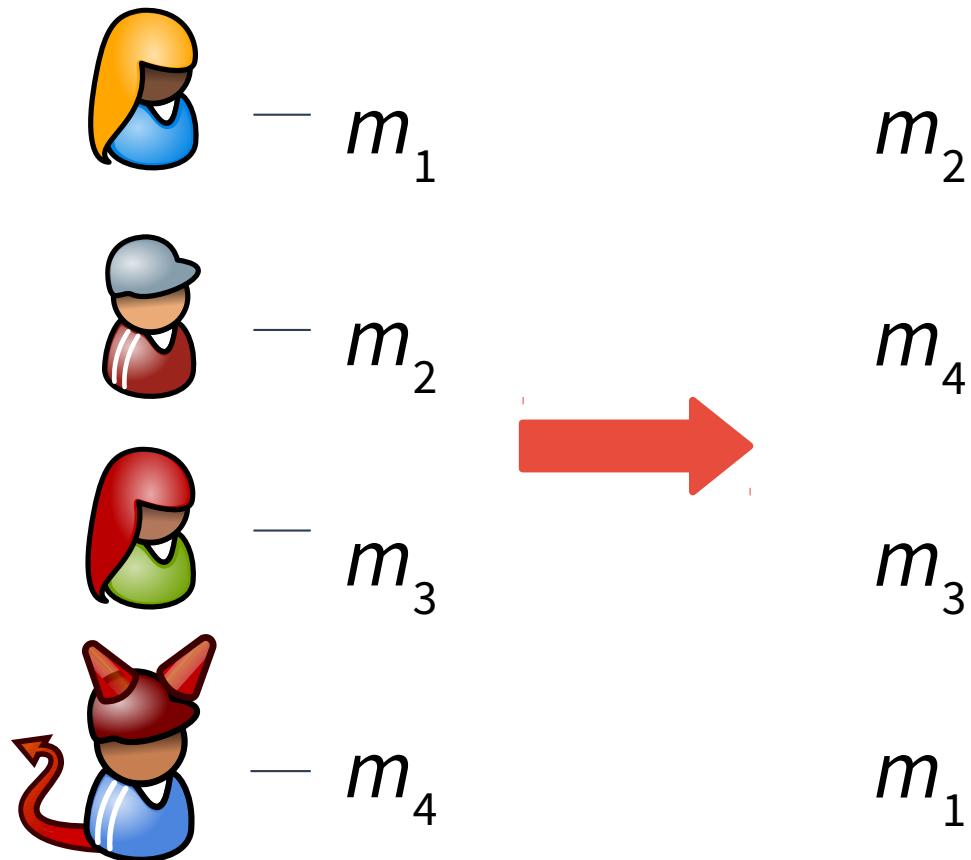


Termination with  
dishonest majority

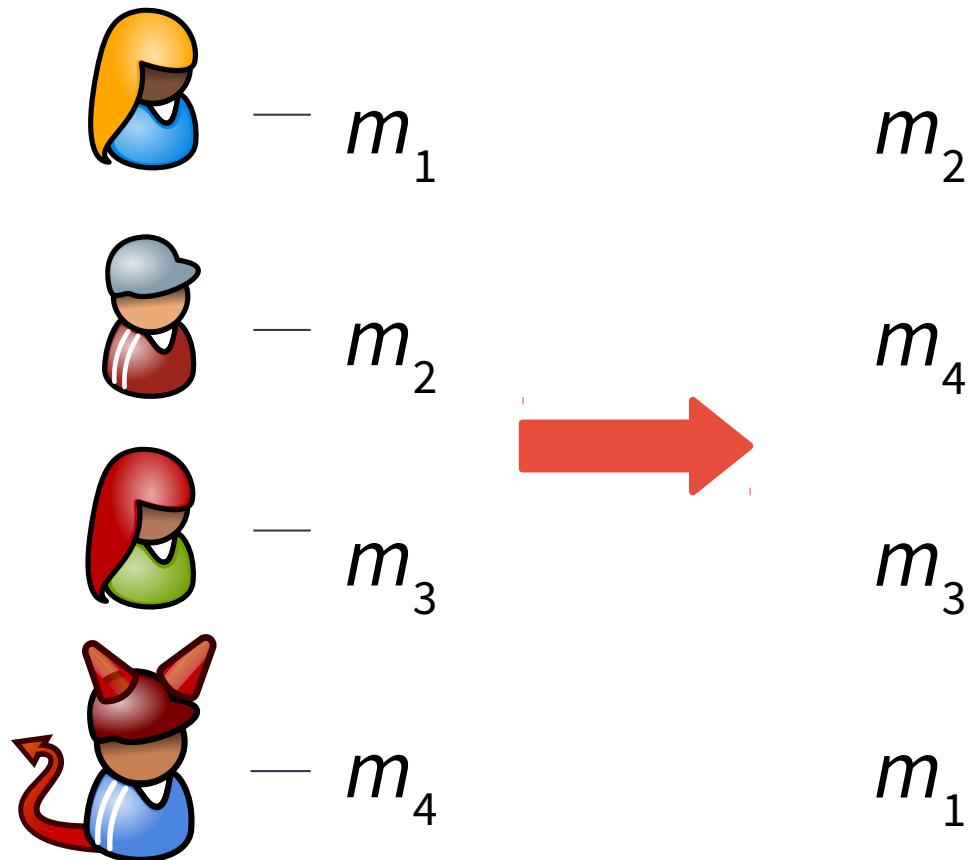
Support for  
fixed messages



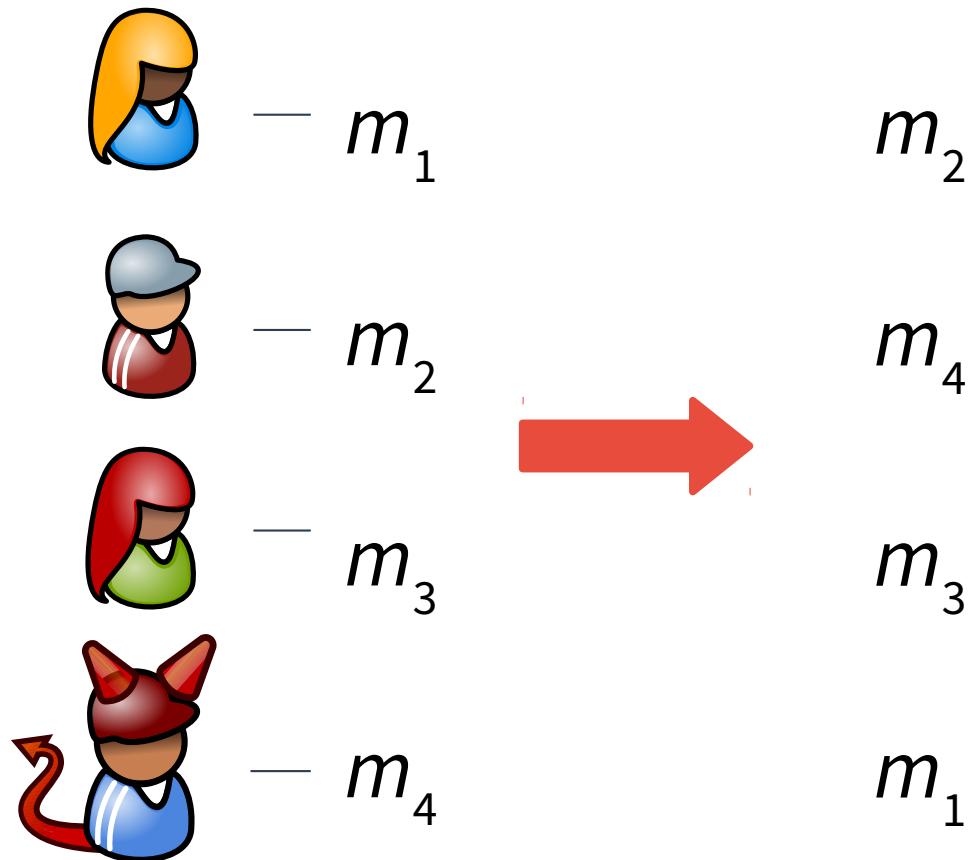
# Idea of Attack on Anonymity



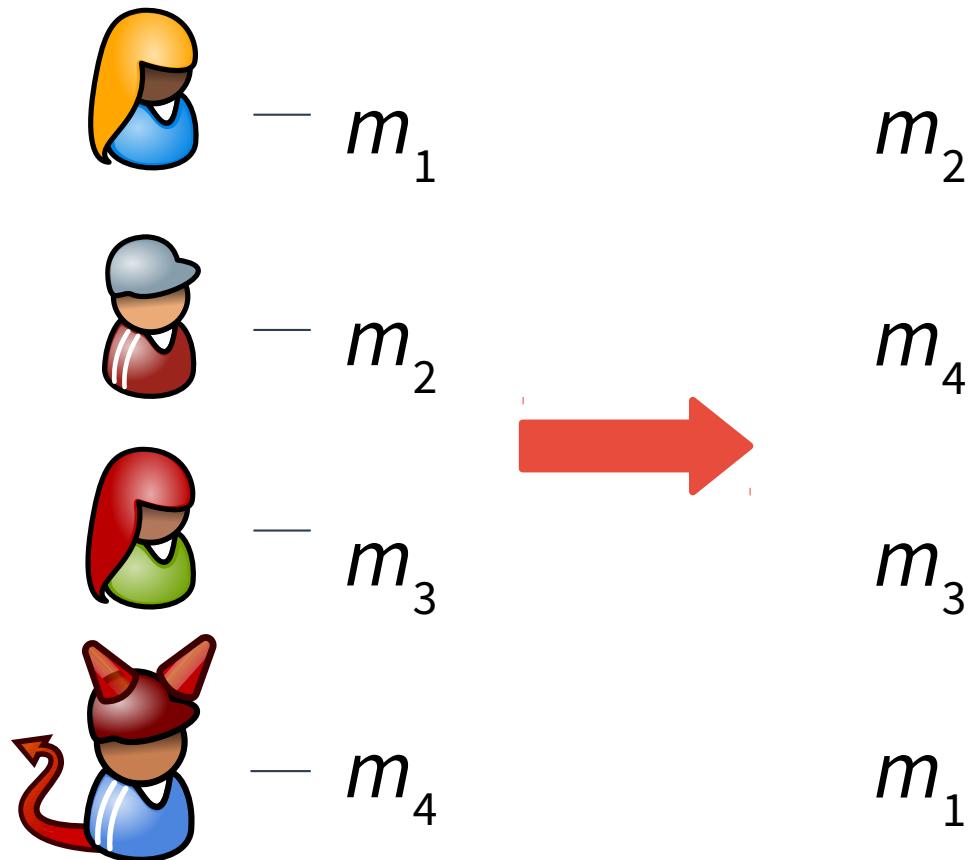
# Idea of Attack on Anonymity



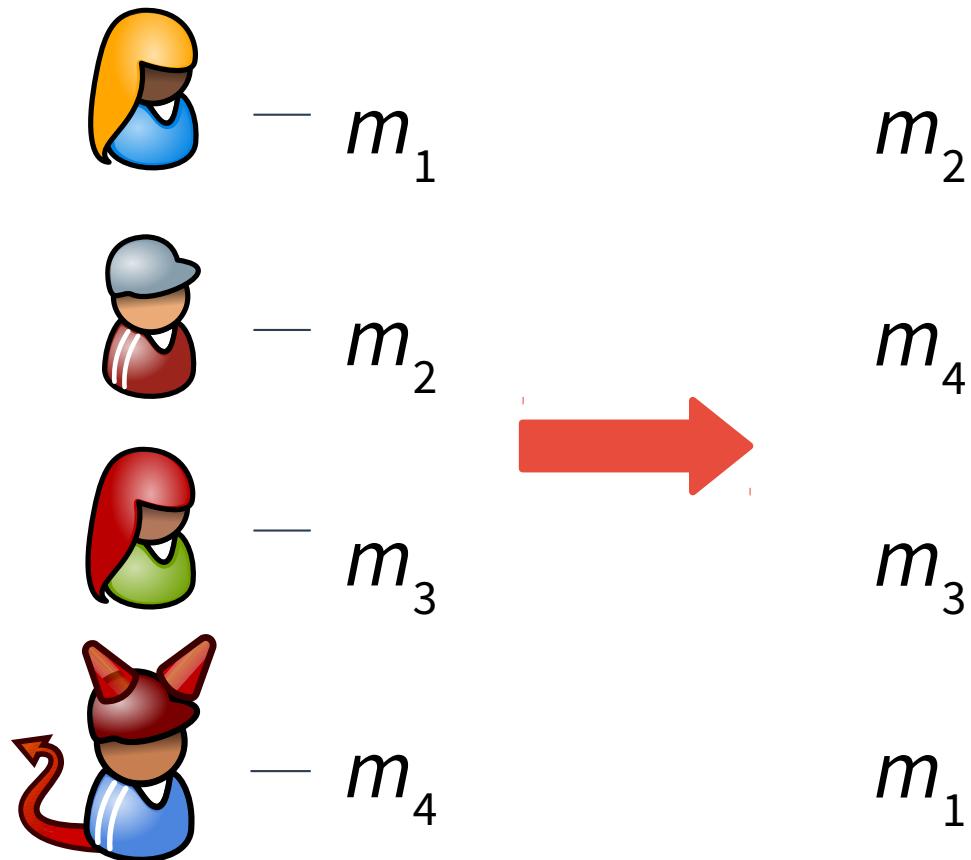
# Idea of Attack on Anonymity



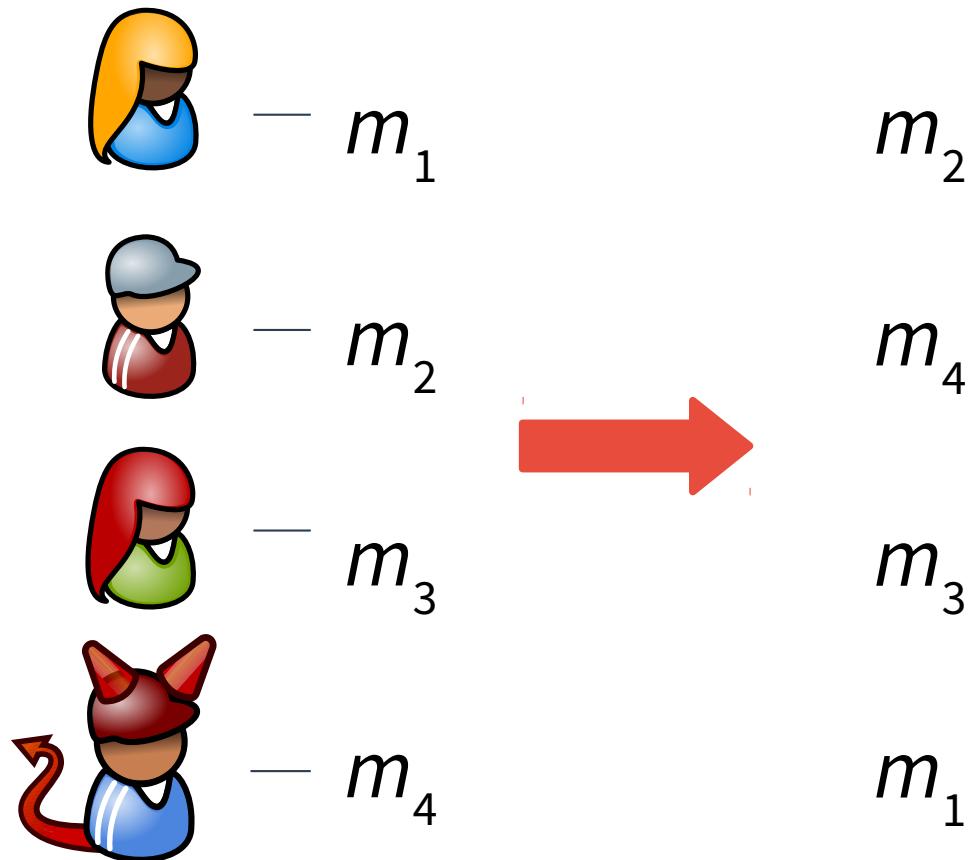
# Idea of Attack on Anonymity



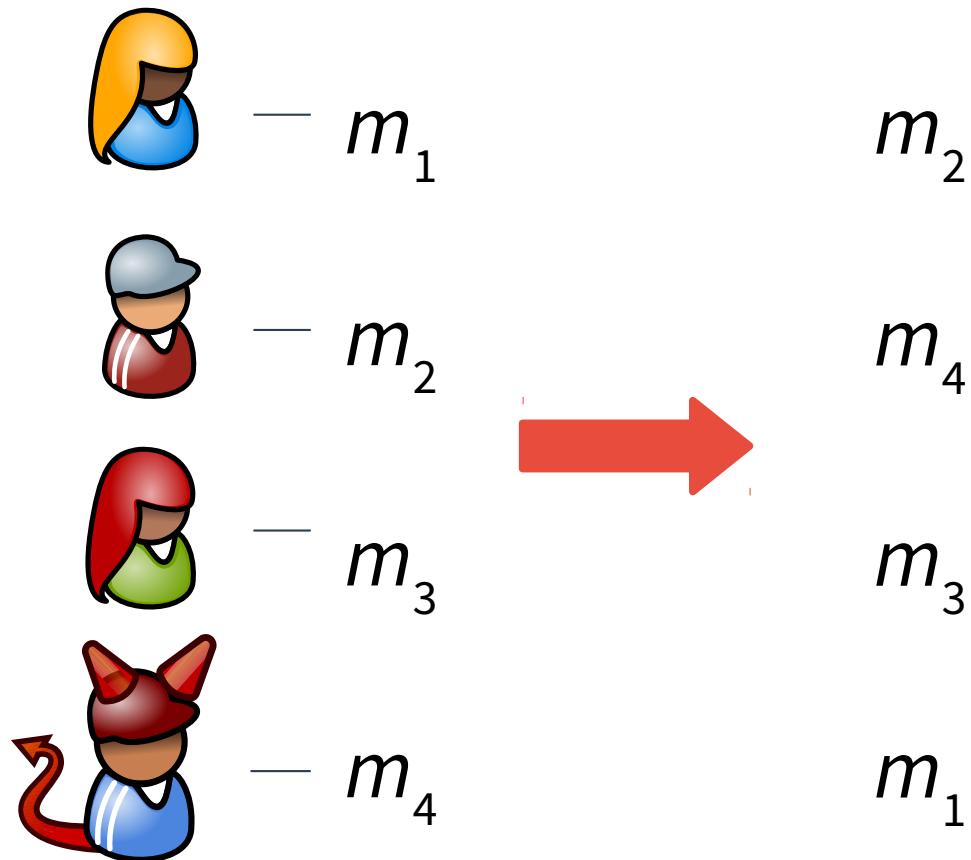
# Idea of Attack on Anonymity



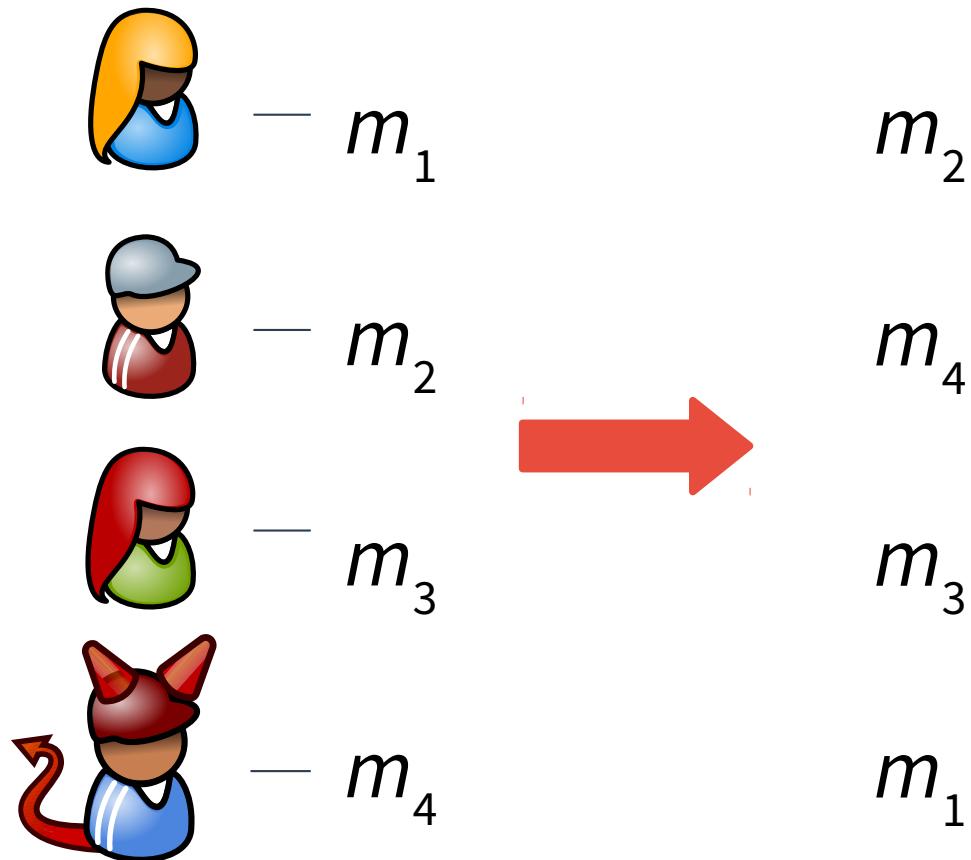
# Idea of Attack on Anonymity



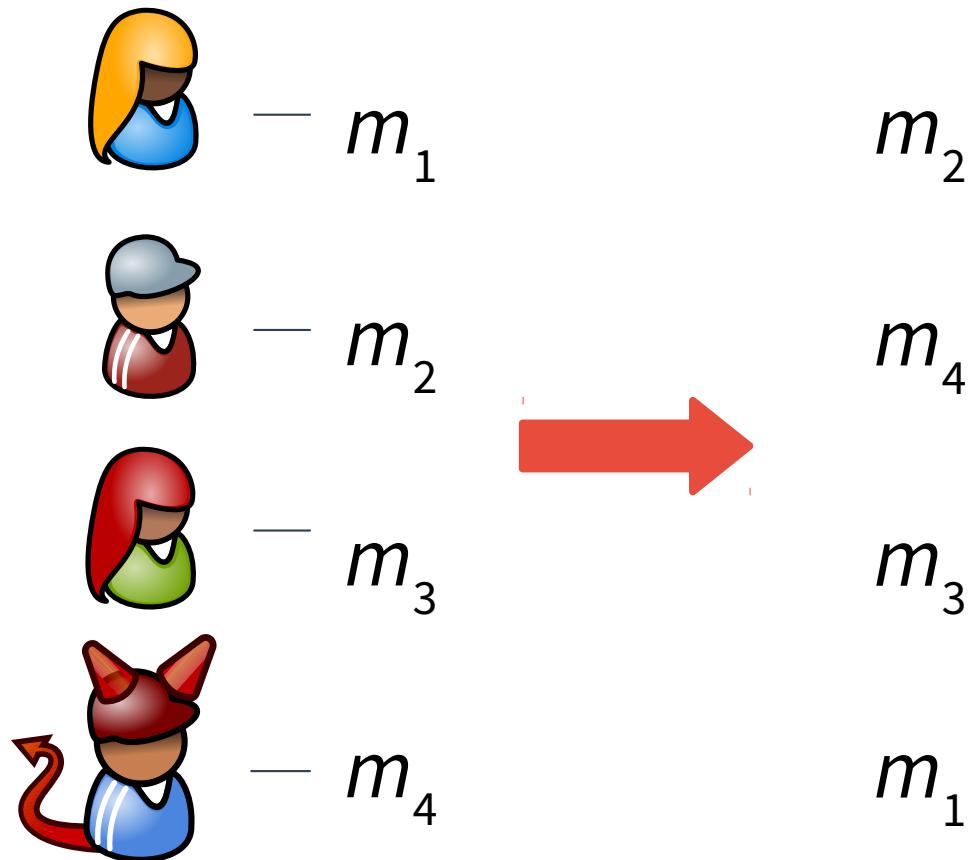
# Idea of Attack on Anonymity



# 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)

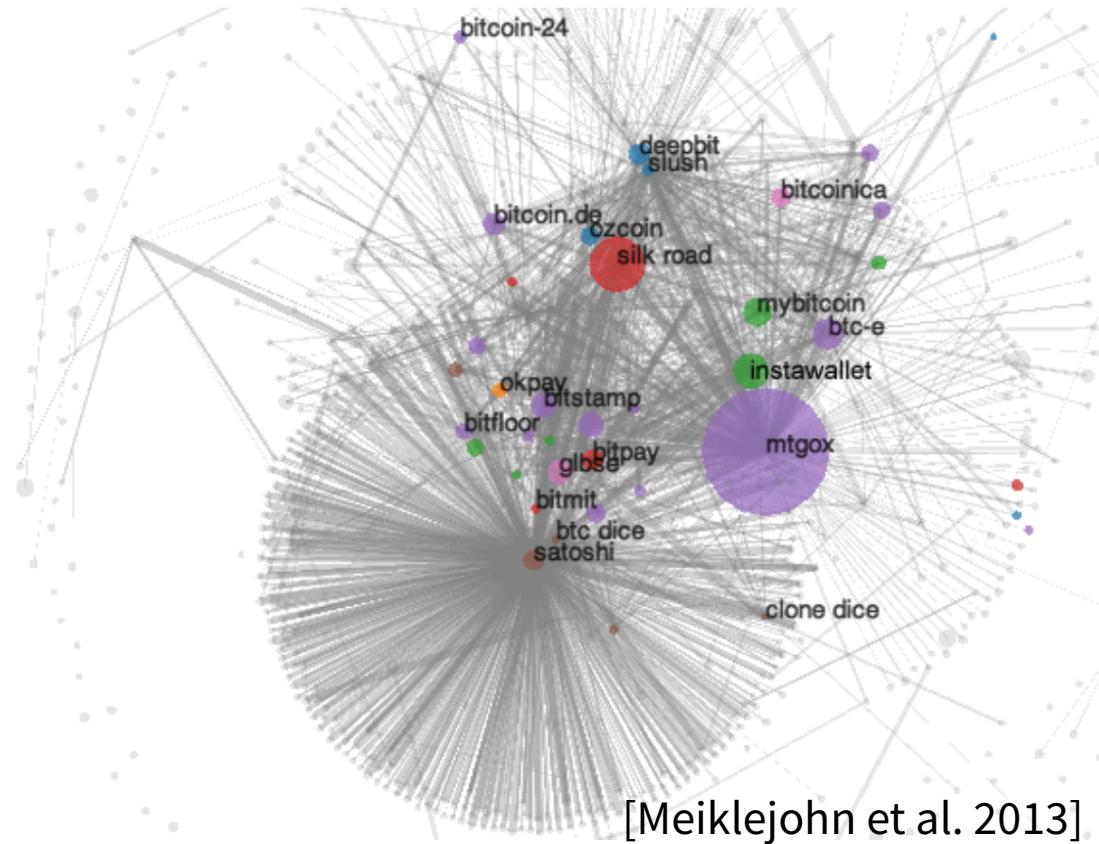
Run 1	KE	CM	DC	SK							
Run 2			KE	CM	DC RV						
Run 3											
(Run 4)											

$4 + 2f$  rounds

# **Transaction Linkability**

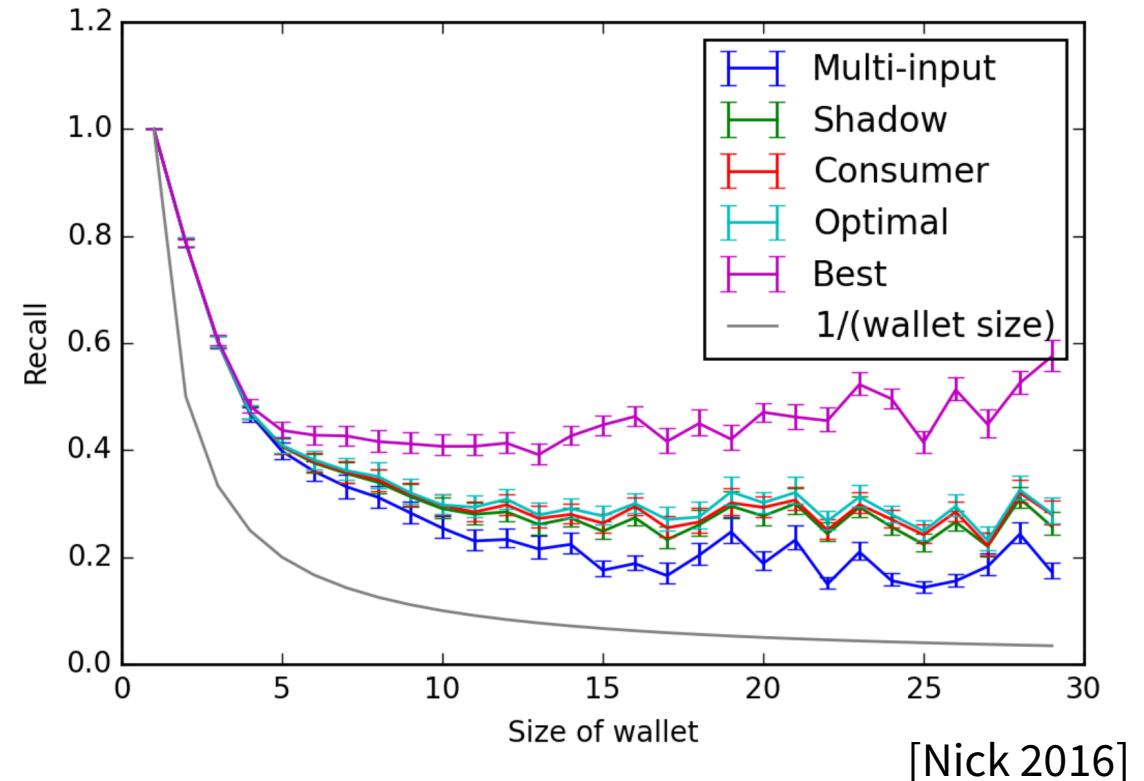
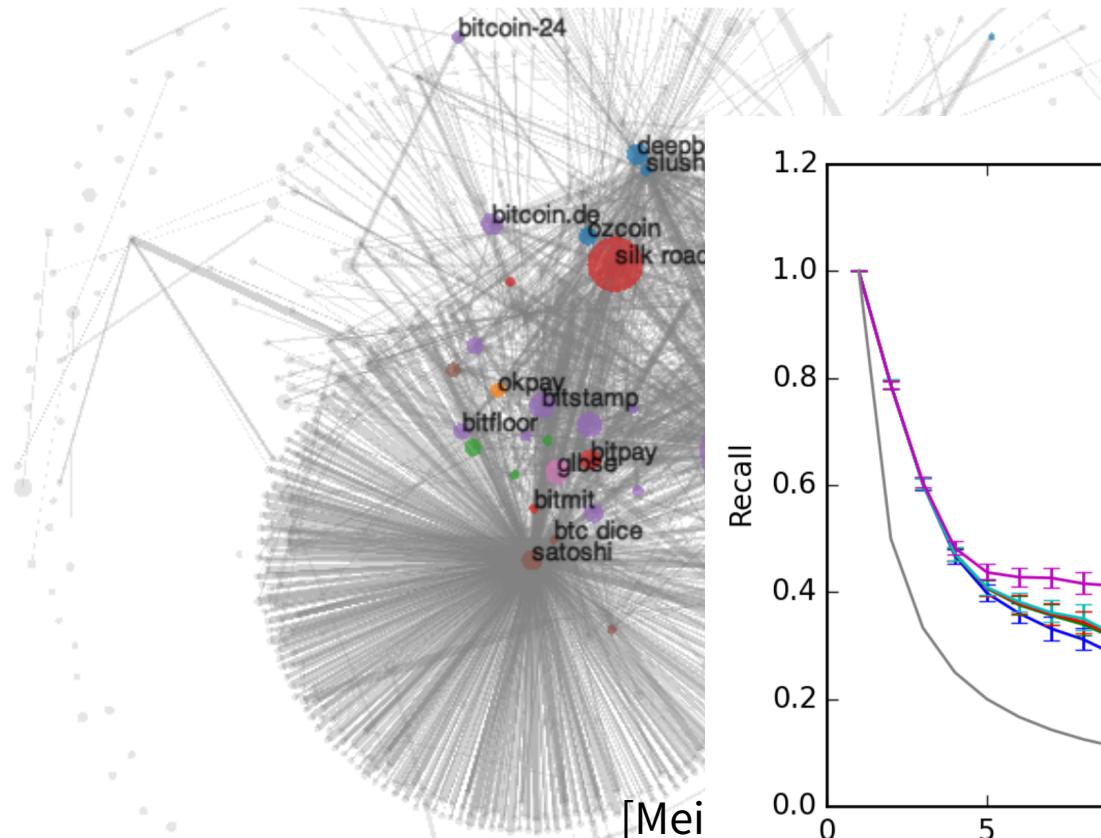
**A Threat to Privacy**

# Linkability and Deanonymization Attacks

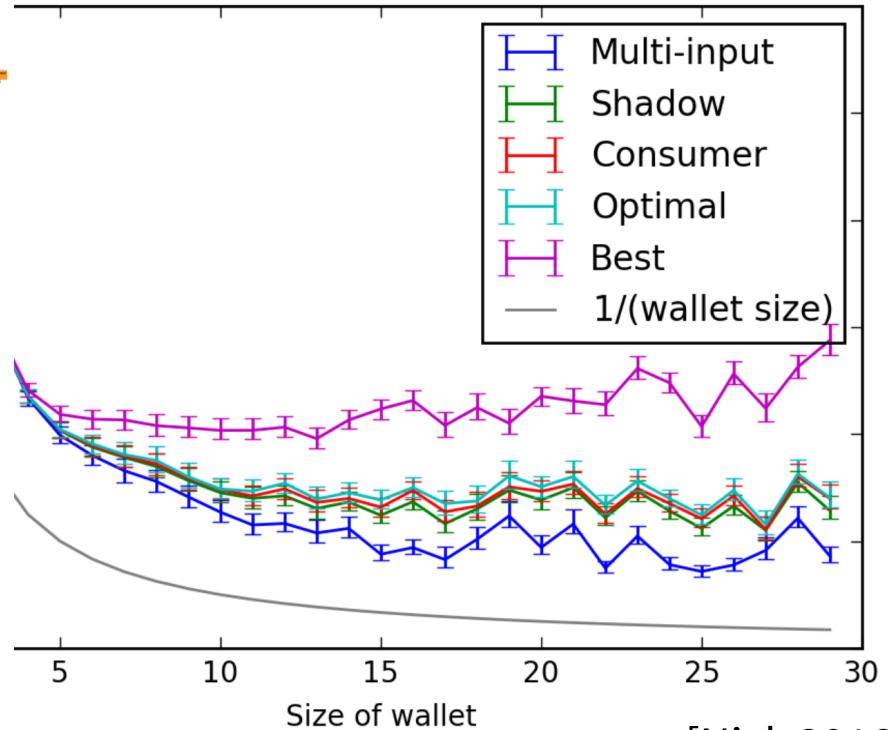
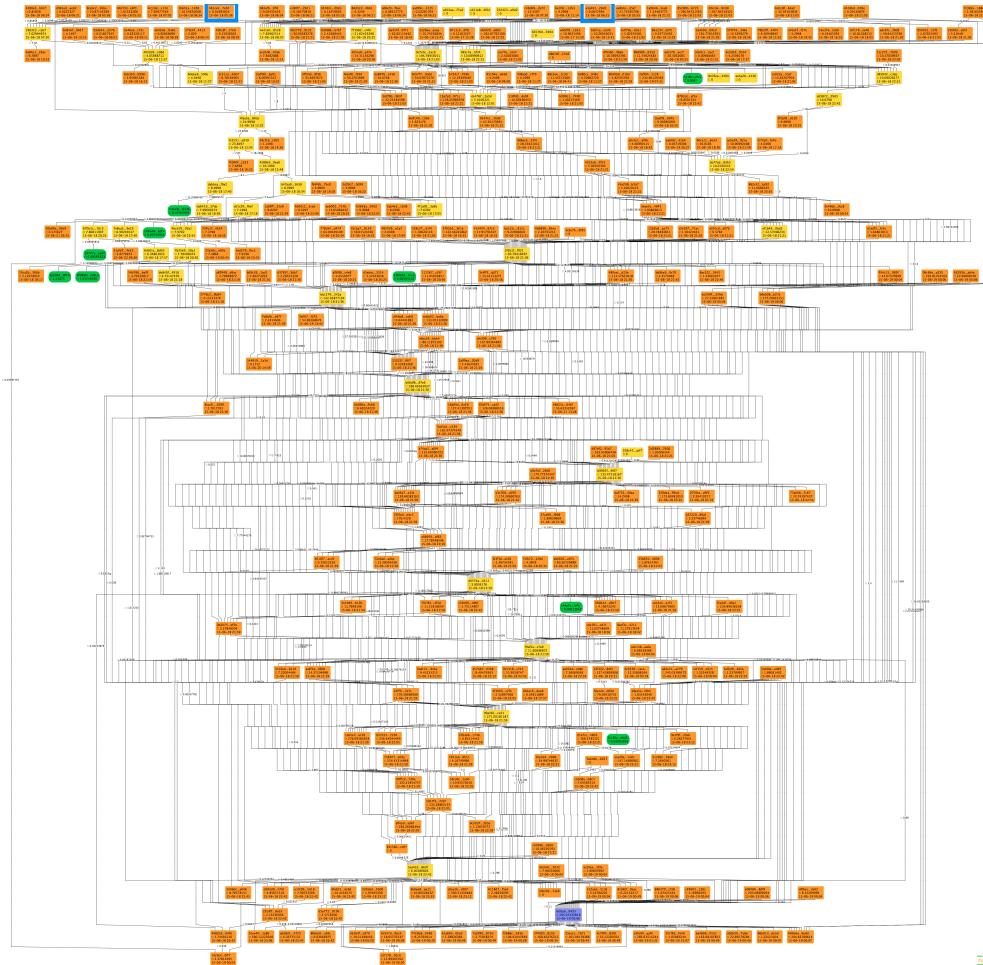


[Meiklejohn et al. 2013]

# Linkability and Deanonymization Attacks



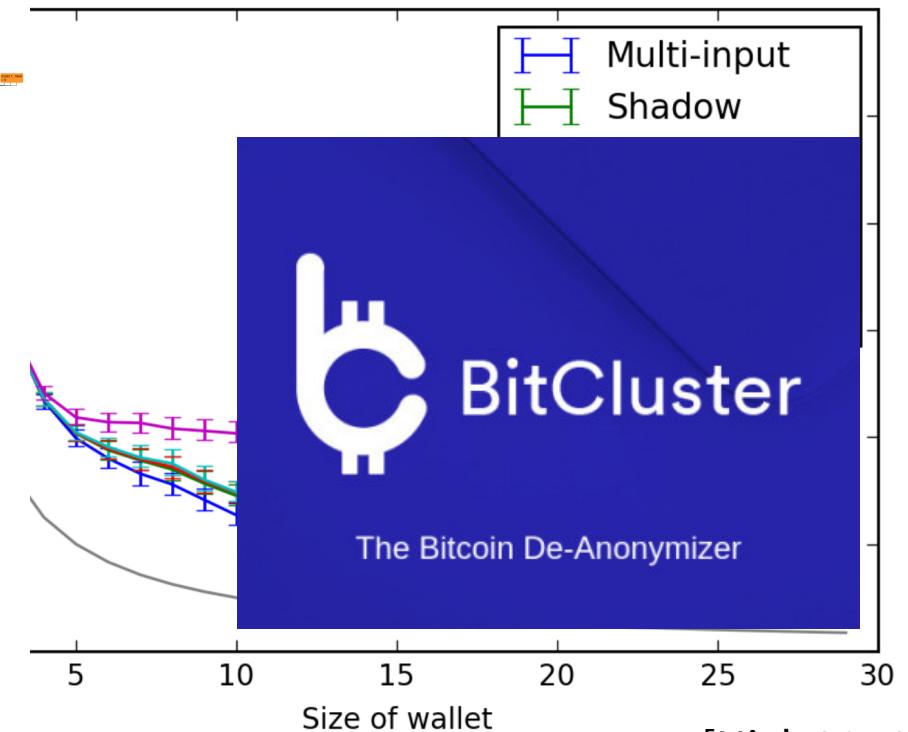
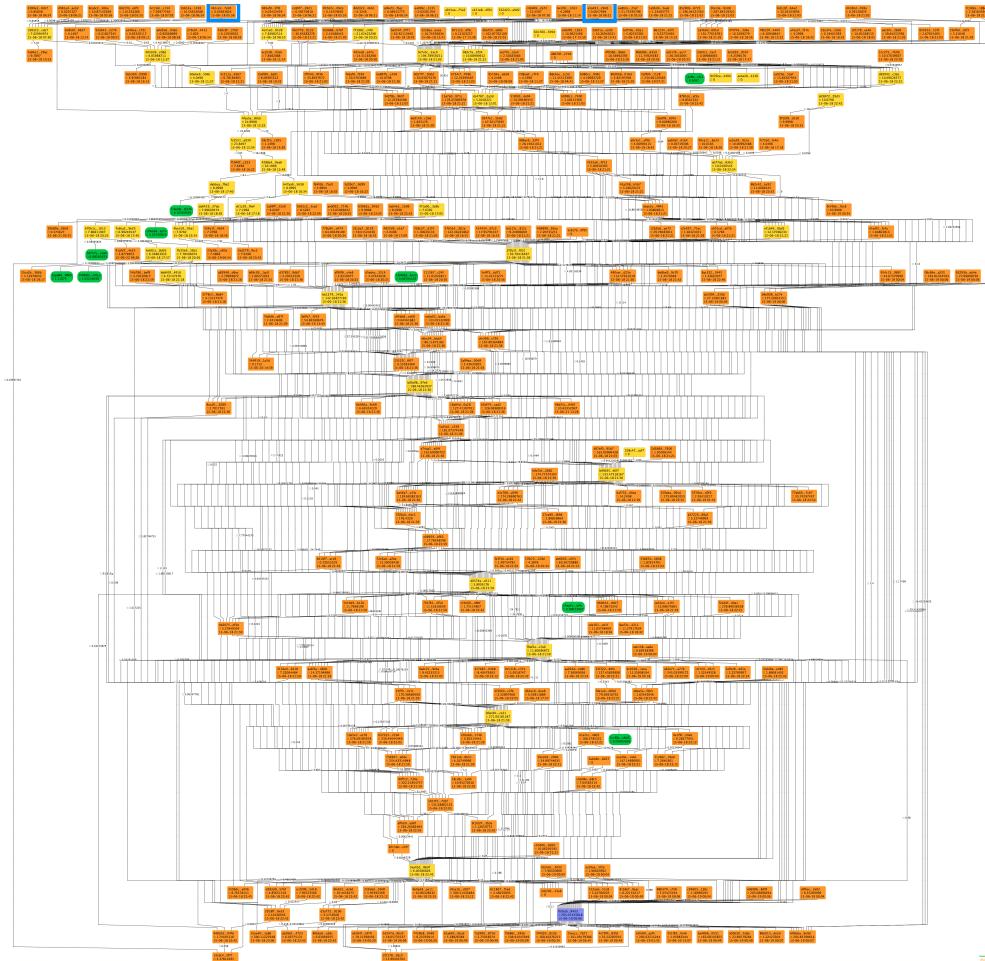
# Linkability and Deanonymization Attacks



[Nick 2016]

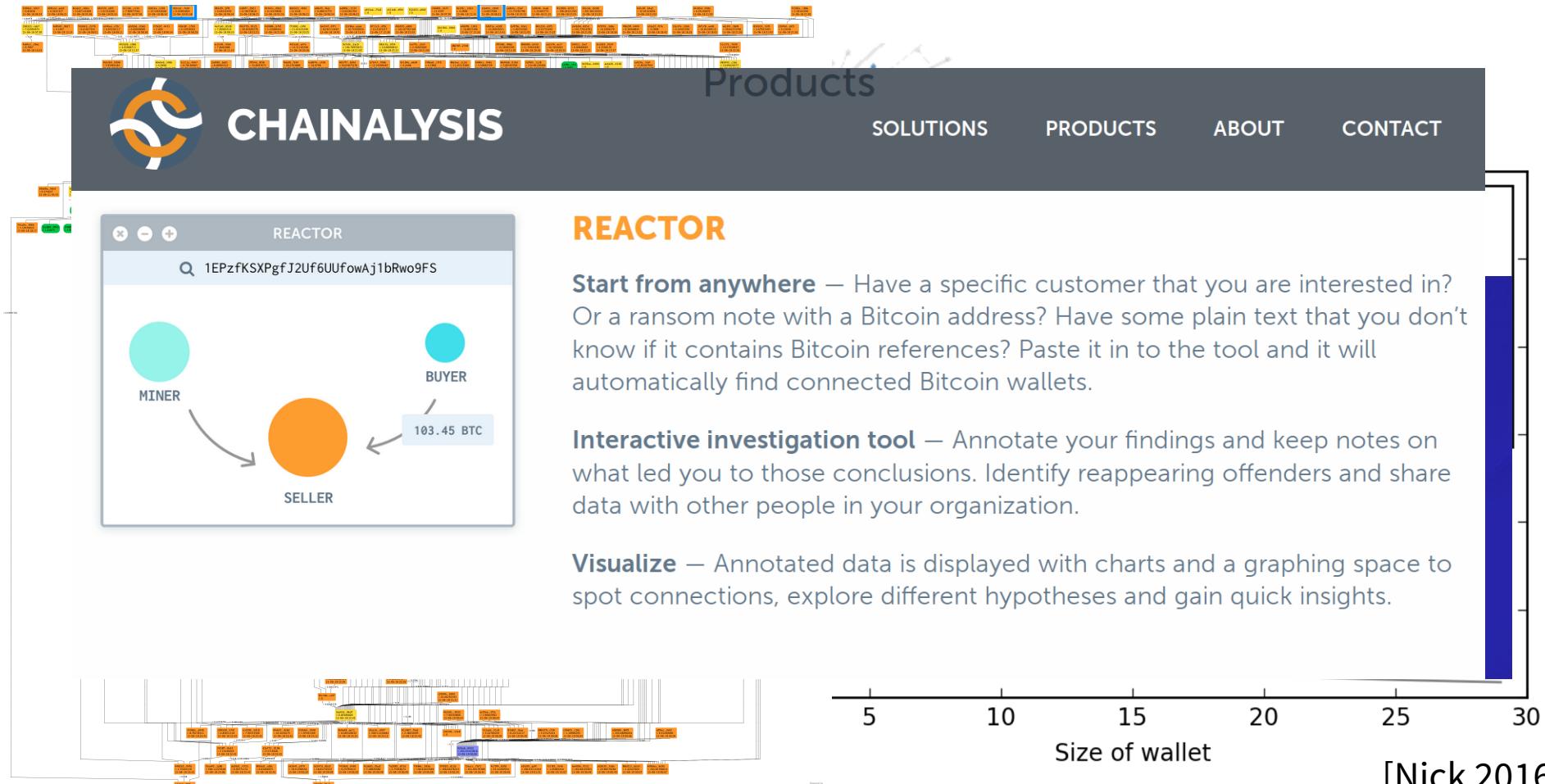
Bitlodine [Spagnuolo, Maggi, Zanero 2013]

# Linkability and Deanonymization Attacks



Bitlodine [Spagnuolo, Maggi, Zanero 2013]

# Linkability and Deanonymization Attacks



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

P2P Mixing and Unlinkable Bitcoin Transactions

Tim Ruffing - @real\_or\_random

# 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

