IT486: Blockchains and Cryptocurrencies

Bitcoin privacy techniques: stealth addresses, mixes, coinjoin

Privacy in payment systems

- Users want privacy in payment systems
- Examples of good uses
  - to avoid being targeted for marketing purposes
  - to avoid discrimination (e.g., for medical conditions)

Privacy in payment systems

- Users want privacy in payment systems
- Examples of good uses
  - to avoid being targeted for marketing purposes
  - to avoid discrimination (e.g., for medical conditions)
- Examples of misuses
  - to prevent detection of bribery
  - to prevent detection of tax evasion

Defining privacy

- What should an adversary be able/unable to learn about the user?
  - their balance
  - whether they are party of a particular transaction? As sender? As receiver?
  - who the counter-parties of their transactions are?
  - what was the amount of the transaction?
- Different systems support different answers to these questions

Psuedonymity

- Pseudonyms are identities (or "fake names") that
  - allow users to engage in events in the system
  - do not allow other users to directly identify the actual user

Psuedonymity

- Pseudonyms are identities (or "fake names") that
  - allow users to engage in events in the system
  - do not allow other users to directly identify the actual user
- Example
  - social media handles, like @satoshi-nakamoto, @bitfinexed are psuedonyms
  - unless the user explicitly wants to be identified, e.g., @realDonaldTrump

Linkability

- Consider that a user u engages in two events e_1 and e_2 in a system
- The two events are linkable by another user v if v can tell that the same user has engaged in both the events (even if v does not know who that user is)

Linkability

- Consider that a user u engages in two events e_1 and e_2 in a system
- The two events are linkable by another user v if v can tell that the same user has engaged in both the events (even if v does not know who that user is)
- Example: @satoshi-nakamoto
  1. posts a message "I have a wonderful new idea: Bitcoin", and later
  2. posts a message "The block reward should be increased"
- An observer can tell that the same person (or group) was responsible for the two events

Linkability is Bad for Privacy

- Suppose we know that a user
  - buys coffee at a cafe every weekday
  - paid ACM membership fees annually the past 20 years
  - bought Andreas Antonopoulos' Understanding Bitcoin on Amazon
- We can probably figure out exactly who they are!

Linkability is Bad for Privacy

- Suppose we know that a user
  - buys coffee at a cafe every weekday
  - paid ACM membership fees annually the past 20 years
  - bought Andreas Antonopoulos' Understanding Bitcoin on Amazon
- We can probably figure out exactly who they are!
- For privacy in a payment system, we want both psuedonymity and unlinkability of payment events

Anonymity set

- To counteract linkability, we can try to hide a user's events amongst other users' events
- Suppose user u engages in some role r (e.g., as sender) in an event e (e.g., a message transmission)
- Suppose an adversary $\mathcal{A}$ observes event e, and other events
- Definition
  - The anonymity set for role r in event e, from the point of view of an adversary $\mathcal{A}$, is the set S of all users v such that $\mathcal{A}$ considers it possible that user v is the user playing role r in e

Anonymity set example

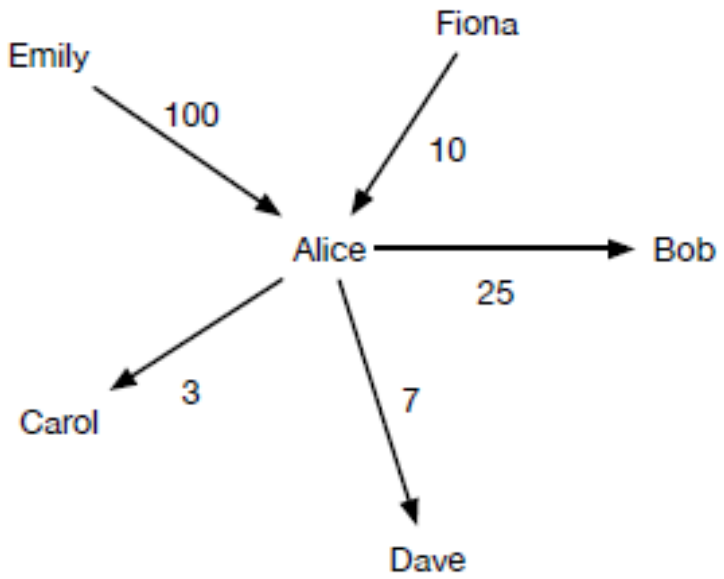
- Each voter u in a set V completes one ballot (not containing their name) and stuffs it into the ballot box
- The ballot box is given a good shake before counting

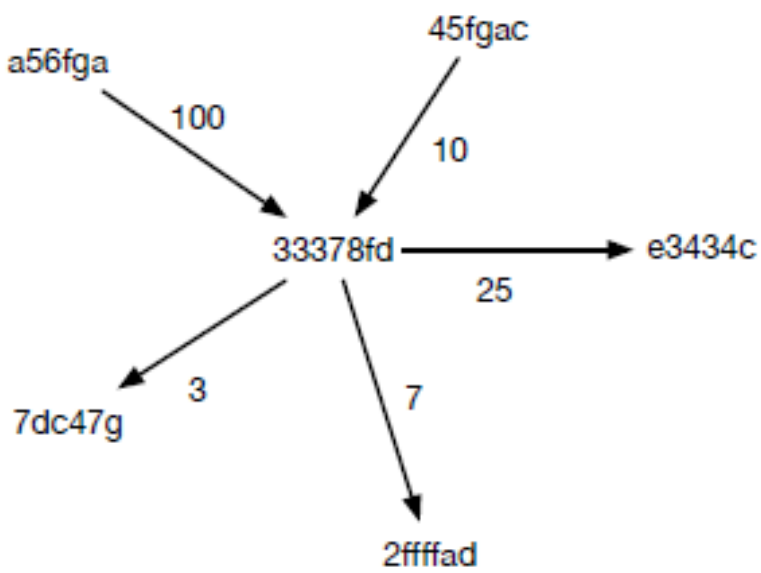
Anonymity set example

- Each voter u in a set V completes one ballot (not containing their name) and stuffs it into the ballot box
- The ballot box is given a good shake before counting
- Assuming the vote counters don't know a priori any voter's intention, the anonymity set for the role voter of each ballot e, from the point of view of the vote counters, is the entire set V

Anonymity in Bitcoin

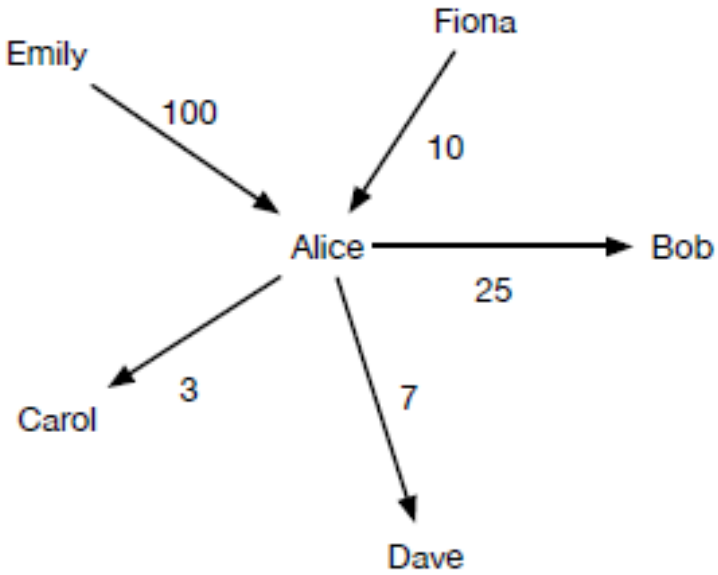
- Suppose Alice transacts as on the left
- If Alice receives money into the same account (33378fd), this looks on the blockchain as on the right (ignoring change)

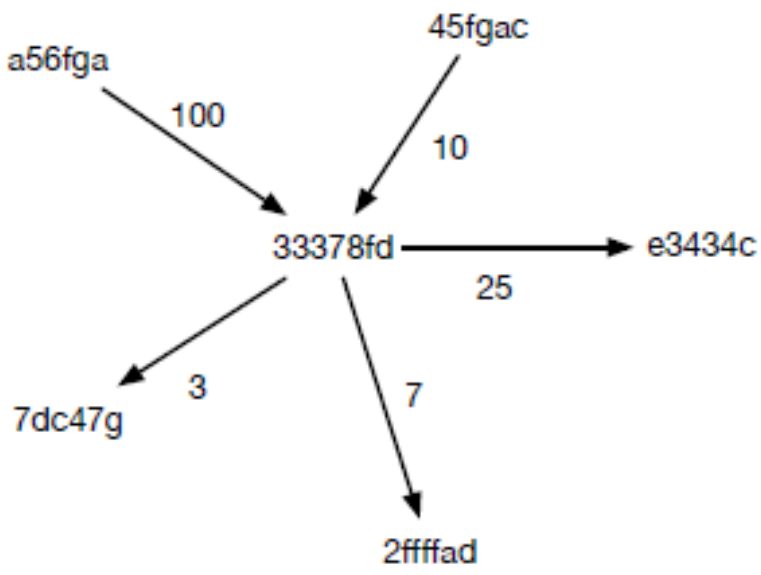




Anonymity in Bitcoin

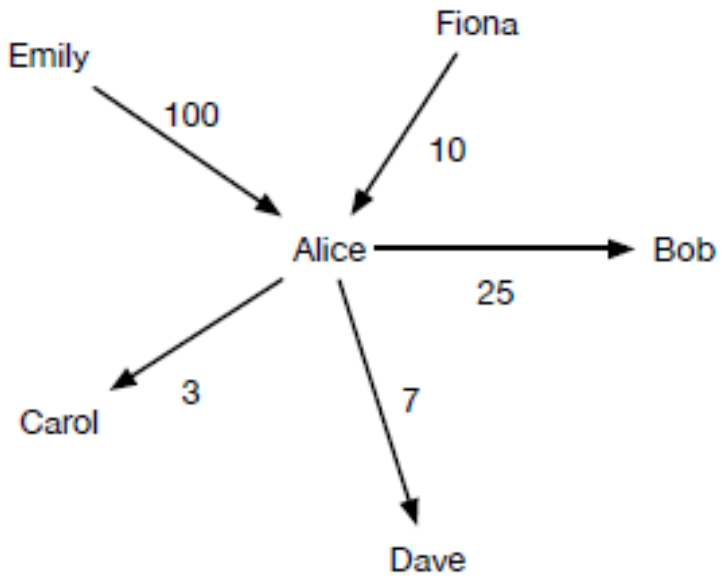
- Assume the real identities of the users of any of these transactions are not known an adversary (other than any of Alice's counter-parties)
- From the adversary's point of view, the anonymity set for the role sender and receiver of any of these transactions is the entire set of users. All the users are psuedonymous

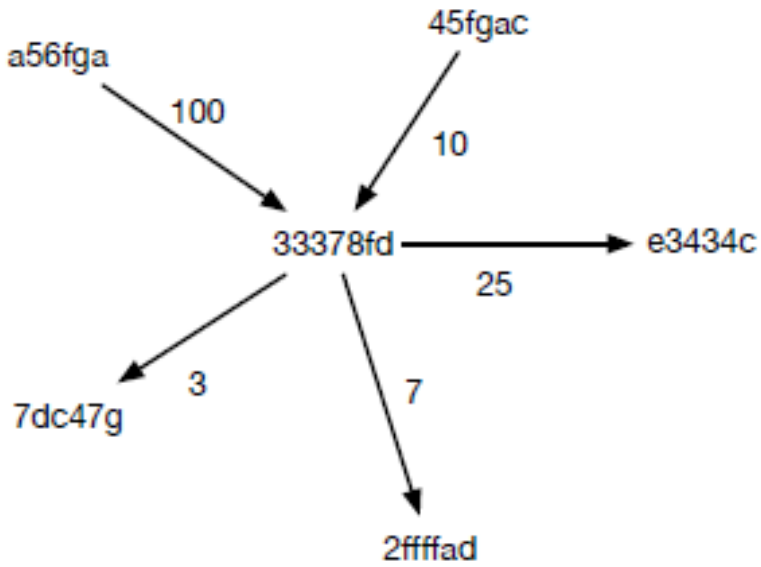




Anonymity in Bitcoin

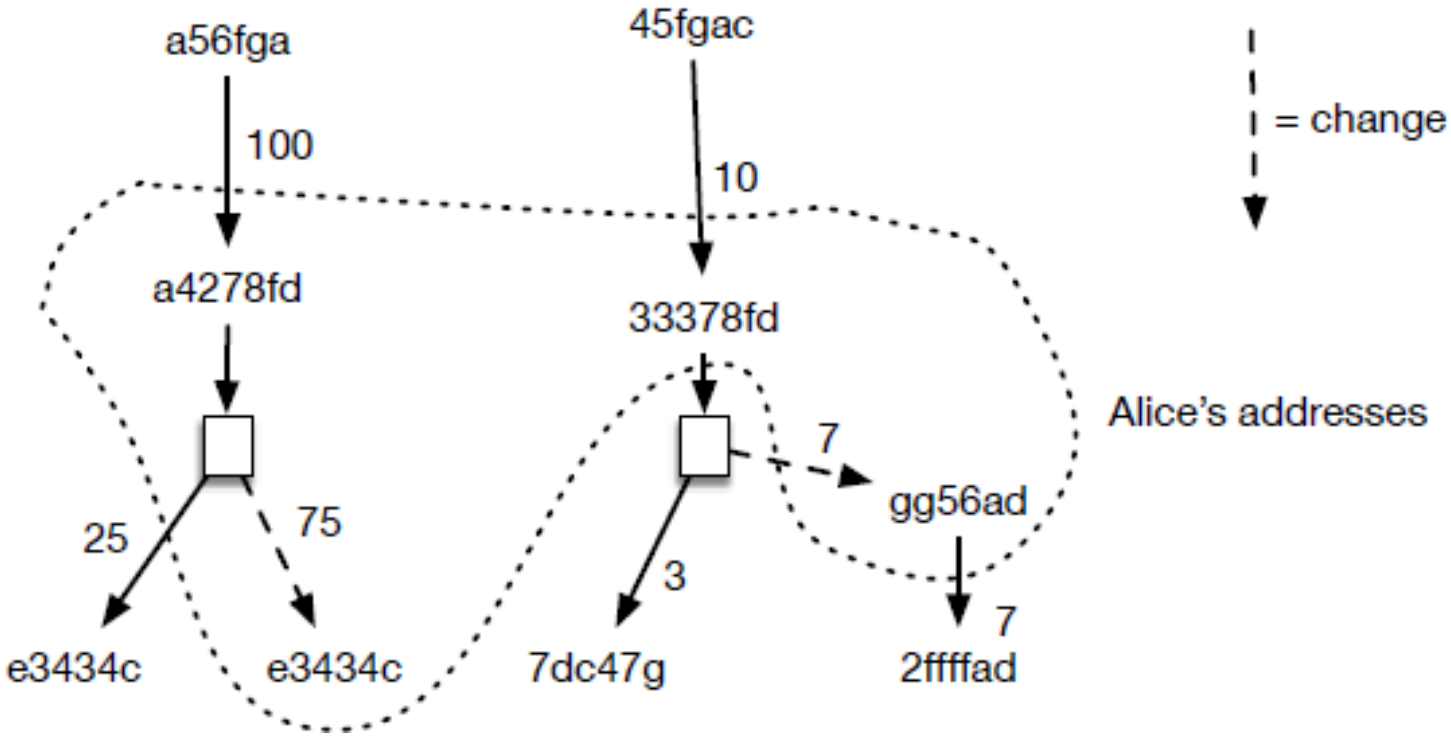
- However, there is substantial linkability of transactions!
- E.g., whoever received 100 also sent 25, whoever received 100 also received 10





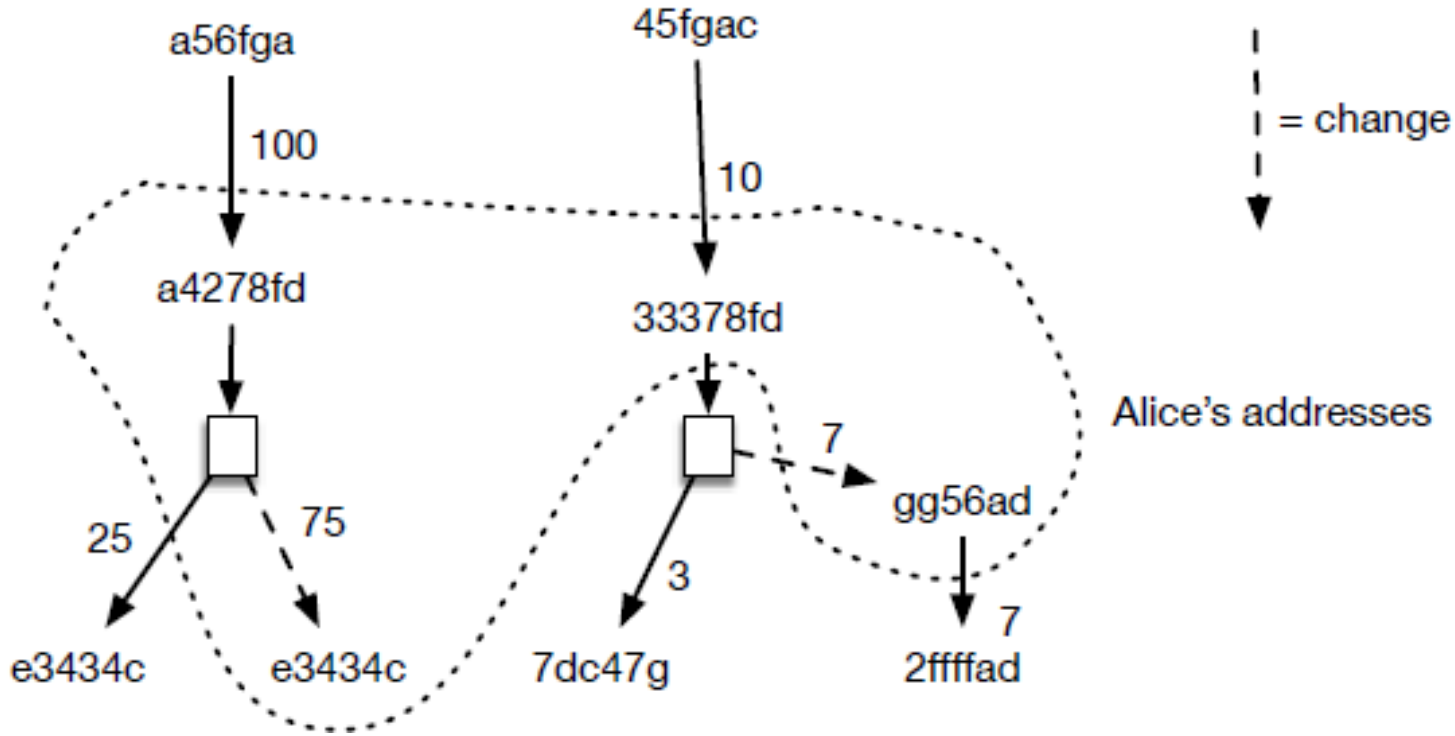
Fresh Addresses

- Alice can decrease the adversary's ability to link transactions by creating a fresh address for each separate amount of money she receives



Fresh Addresses

- Alice can decrease the adversary's ability to link transactions by creating a fresh address for each separate amount of money she receives



- Note that this unlinks some of Alice's transactions, but still leaves links between Alice's incoming and outgoing money!

Diffie-Hellman key exchange for elliptic curves

Alice and Bob agree on an elliptic curve E and a point on it P (chosen carefully). The choice (E, P) is public.

Alice chooses a secret integer a, computes $Q_A = aP$ and sends Q_A to Bob
Bob chooses a secret integer b, computes $Q_B = bP$ and sends Q_B to Alice

- Alice computes $aQ_B = abP$. Bob computes $bQ_A = abP$.
- They agree of a way to extract a key from abP.

Diffie-Hellman Problem

- Given E, P, $Q_A(= aP)$, $Q_B(= bP)$, compute abP
  - this is known as the Elliptic Curve Computational Diffie-Hellman Problem (CDHP)
- Presumably, solving this problem requires solving the elliptic curve discrete logarithm problem
- If you can solve DLP then you can solve CDHP, but the converse is not proven!

Stealth addresses

- Bob maintains one pre-generated public address
- To send money to Bob, Alice generates a one-time key based on Bob's public address
- Bob monitors the blockchain for payments
- Bob can recognize payments to one-time keys from his address using his private key

Stealth addresses

- Bob maintains one pre-generated public address
- To send money to Bob, Alice generates a one-time key based on Bob's public address
- Bob monitors the blockchain for payments
- Bob can recognize payments to one-time keys from his address using his private key
- Mallory cannot distinguish whether a payment belongs to Bob

Stealth addresses

- Bob can now publish his stealth address to everybody
- Each output sent to Bob will look to observers as having different destinations
- Nobody can tell these outputs are going to Bob
- Nobody can tell these outputs are going to the same person

Stealth addresses

- Bob creates two EC key pairs (A, a) and (B, b)
- (a, b) is his private key
- a is the view key (or tracking key)
- b is the spending key
- Bob publishes (A, B) as his public key

Send money to stealth address

- G is elliptic curve base point
- H is hash function
- Alice wants to pay Bob to (A, B)
- She generates random r and publishes $R = rG$
- Computes one-time public key $P = H(rA)G + B$
- Alice sends her funds to Bob using the following locking script
  - `P CHECK_SIG`

Viewing money on stealth address

- For every transaction on the blockchain, Bob computes $P' = H(aR)G + B$
- Bob checks if $P = P'$

Viewing money on stealth address

- For every transaction on the blockchain, Bob computes $P' = H(aR)G + B$
- Bob checks if $P = P'$
- $P' = H(aR)G + B$

$$= H(arG)G + B$$
$$= H(raG)G + B$$
$$= H(rA)G + B$$
$$= P$$

Viewing money on stealth address

- For every transaction on the blockchain, Bob computes $P' = H(aR)G + B$
- Bob checks if $P = P'$
- $P' = H(aR)G + B$

$$= H(arG)G + B$$
$$= H(raG)G + B$$
$$= H(rA)G + B$$
$$= P$$

- Only a is needed to view money; a is a view key

Spending money from stealth address

- To spend the money, Bob needs to know x such that $P = xG$
- Can Bob compute x?

Spending money from stealth address

- To spend the money, Bob needs to know x such that $P = xG$
- Can Bob compute x?
- Set $x = H(aR) + b$
- $xG = (H(aR) + b)G$

$$= H(aR)G + bG$$
$$= H(aR)G + B$$
$$= P$$

Spending money from stealth address

- To spend the money, Bob needs to know x such that $P = xG$
- Can Bob compute x?
- Set $x = H(aR) + b$
- $xG = (H(aR) + b)G$

$$= H(aR)G + bG$$
$$= H(aR)G + B$$
$$= P$$

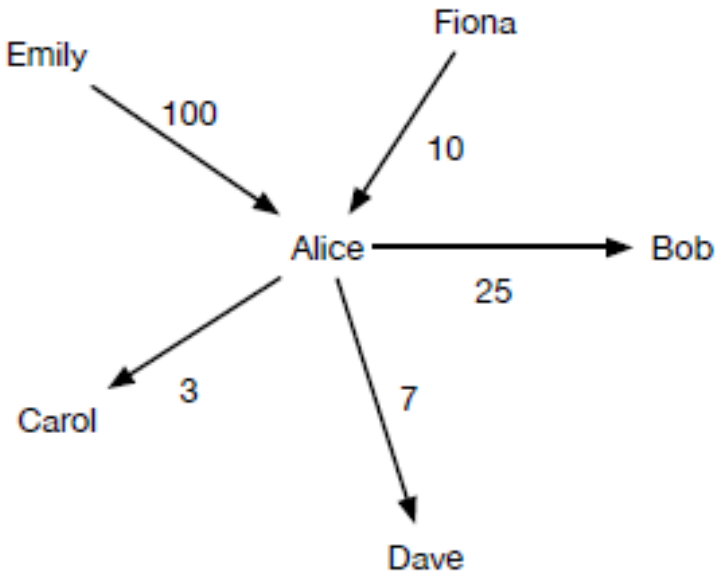
Spending money from stealth address

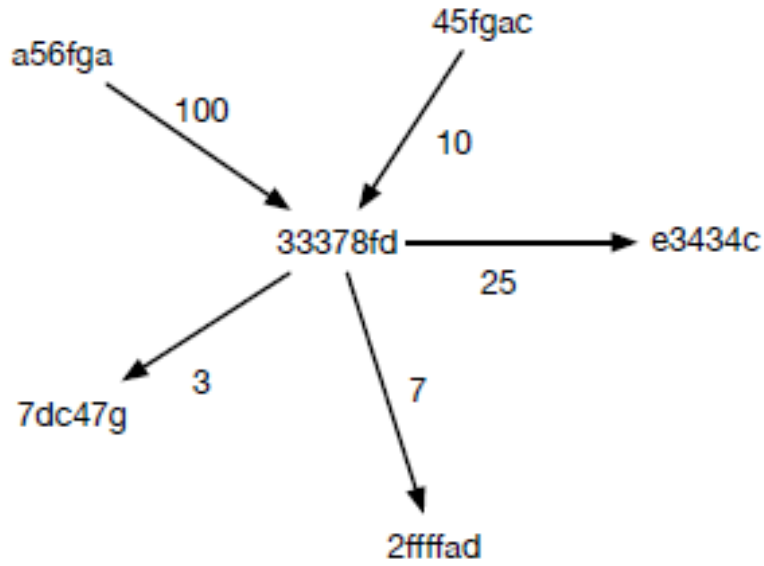
- To spend the money, Bob needs to know x such that $P = xG$
- Can Bob compute x?
- Set $x = H(aR) + b$
- $xG = (H(aR) + b)G$

$$= H(aR)G + bG$$
$$= H(aR)G + B$$
$$= P$$

- b is needed to spend money; b is a *spending key*

Counter-party viewpoint

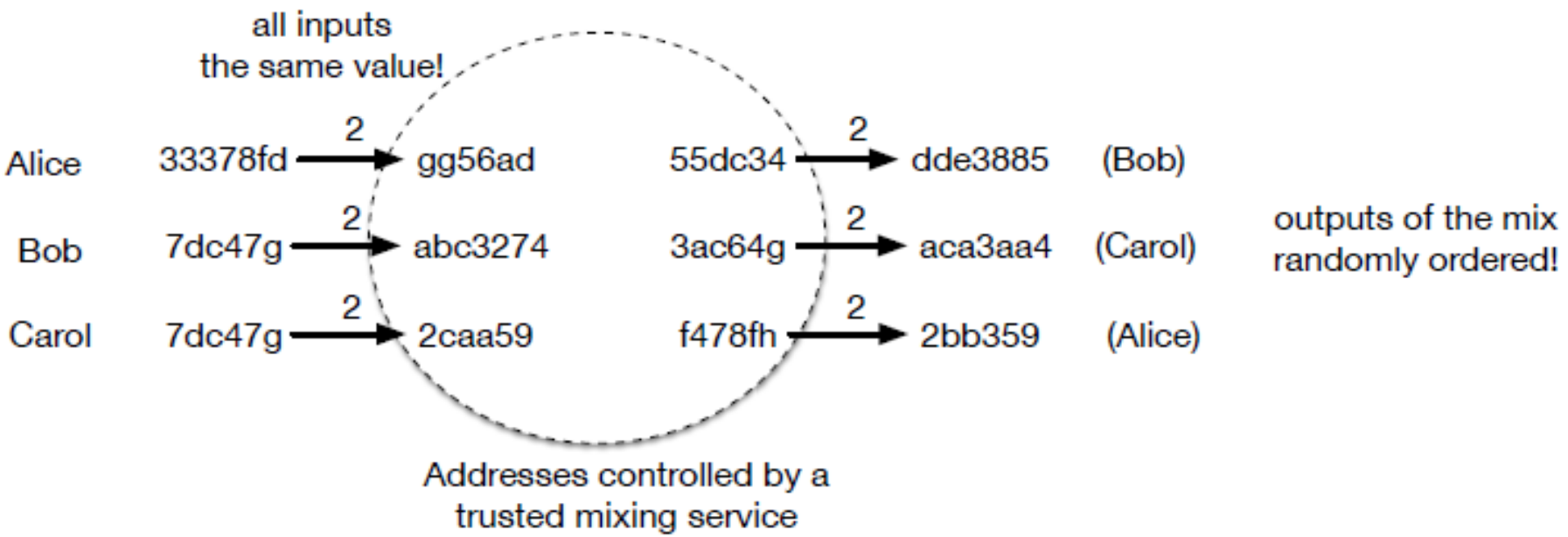




- If Emily knows that Alice was the recipient of her 100, then from the point of view of Emily:
  - the anonymity set of role "sender" in each of the transactions 3, 7, 25 is $\{Alice\}$

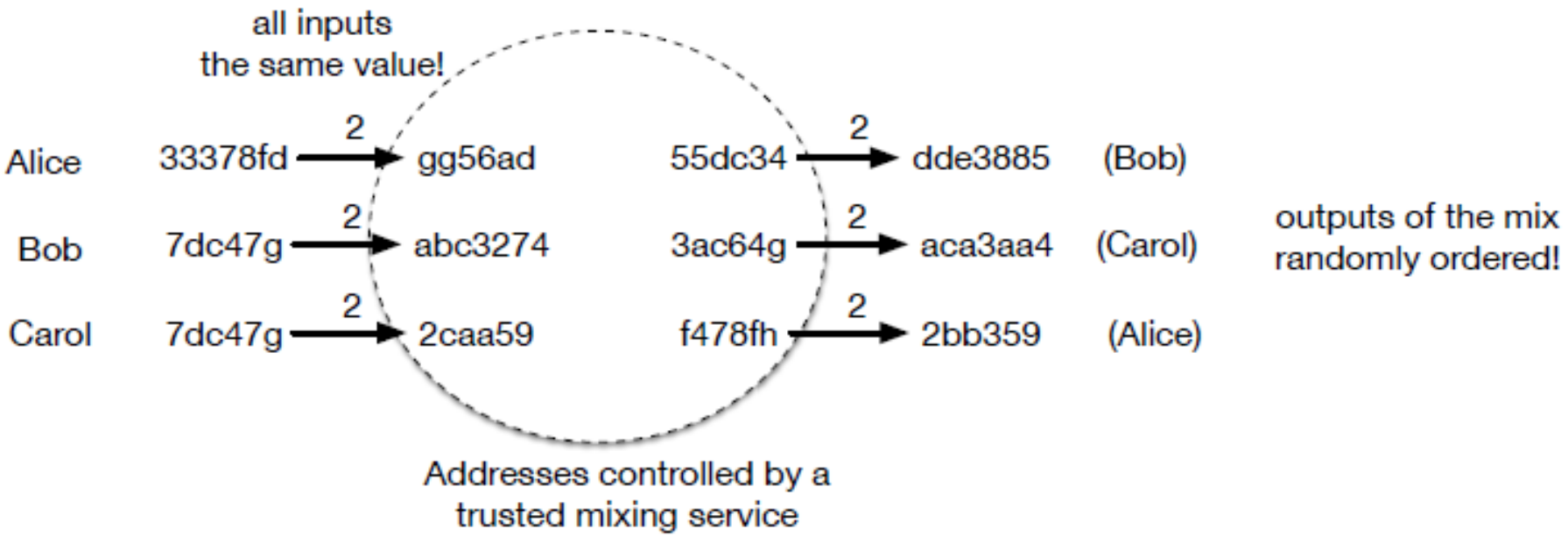
Mixing

- To hide what you are doing with your money from a user who knows your address, you would like to increase the size of the anonymity set with respect to that user
- One way is to "mix" your money with that of a set of other users



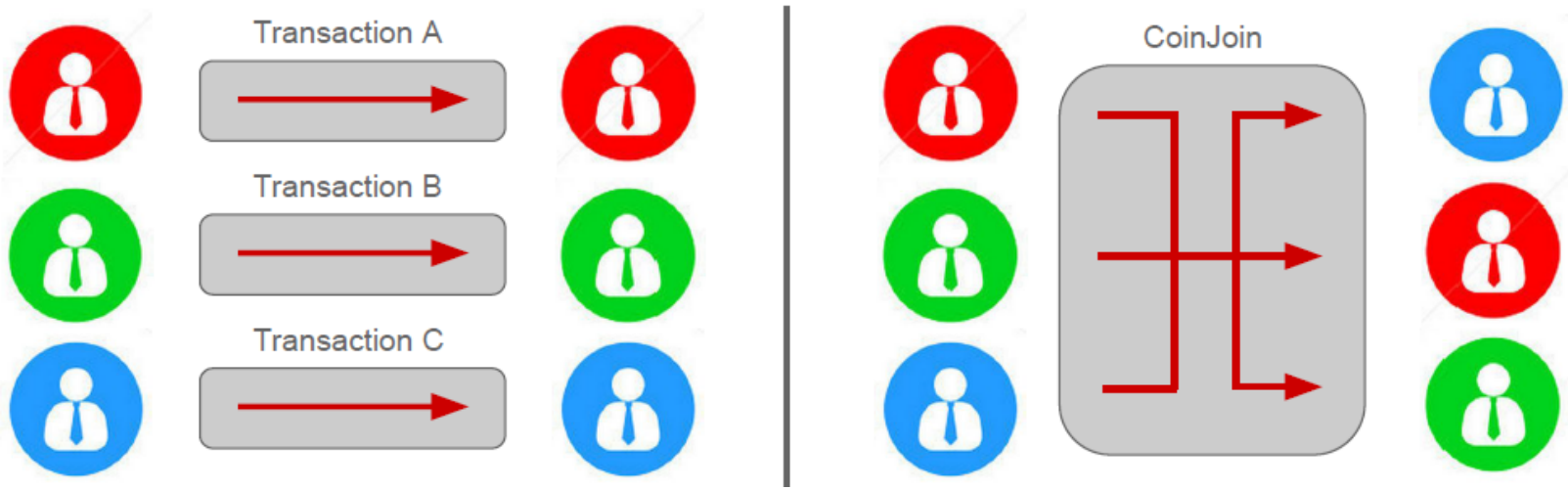
Mixing

- Risk: the mixing service is trusted and could steal your money!



CoinJoin

- Proposed by gmaxwell in 2013
- Inputs and outputs for multiple transactions combined into a single Bitcoin transaction.



CoinJoin steps

- Step 1: Participants send inputs, outputs and amounts to a facilitator.
- Step 2: Facilitator combines inputs and outputs into a single Bitcoin transaction.
- Step 3: Transaction is passed to each participant for signing.
- Step 4: Fully signed transaction is broadcast.

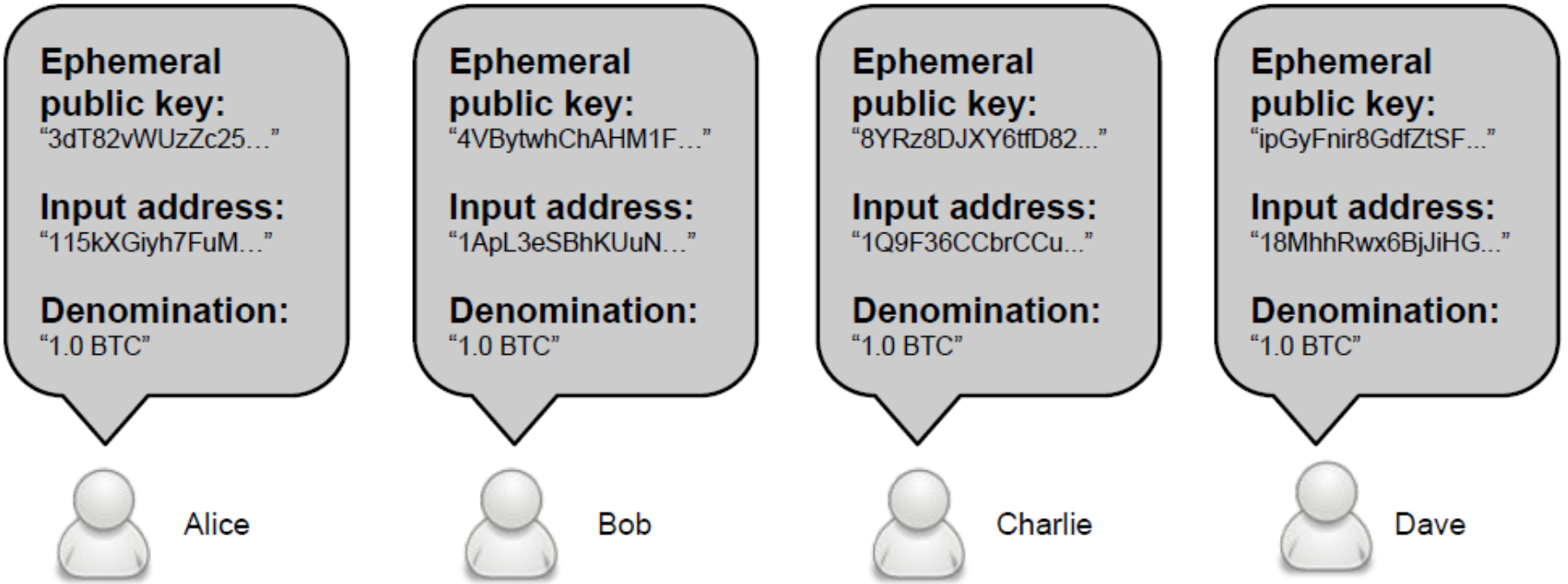
Features of CoinJoin

- No changes to Bitcoin protocol necessary.
- Requires a trusted party to organize.
- Trusted party has no ability to steal coins (unlike traditional mixer), but can know the mapping of inputs and outputs.

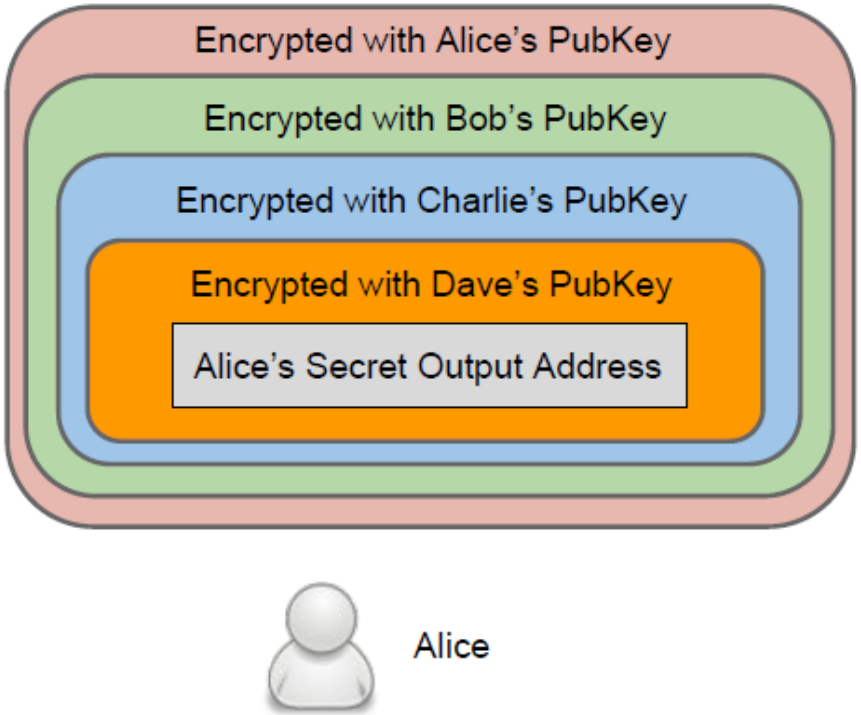
CoinShuffle

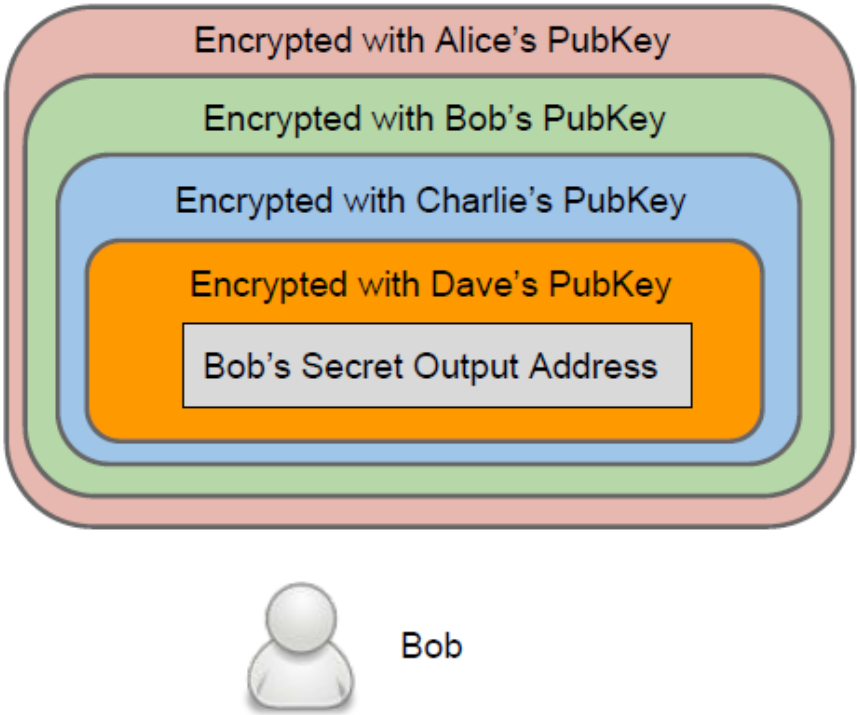
- Proposed in 2014 by Ruffing, Moreno-Sanchez and Kate
- Uses underlying CoinJoin transaction to get unlinkability, requires no changes to Bitcoin protocol.
- Additionally: No participant or central party needs to be trusted, nobody has access to the mapping between inputs and outputs

CoinShuffle Step 1: Announcement

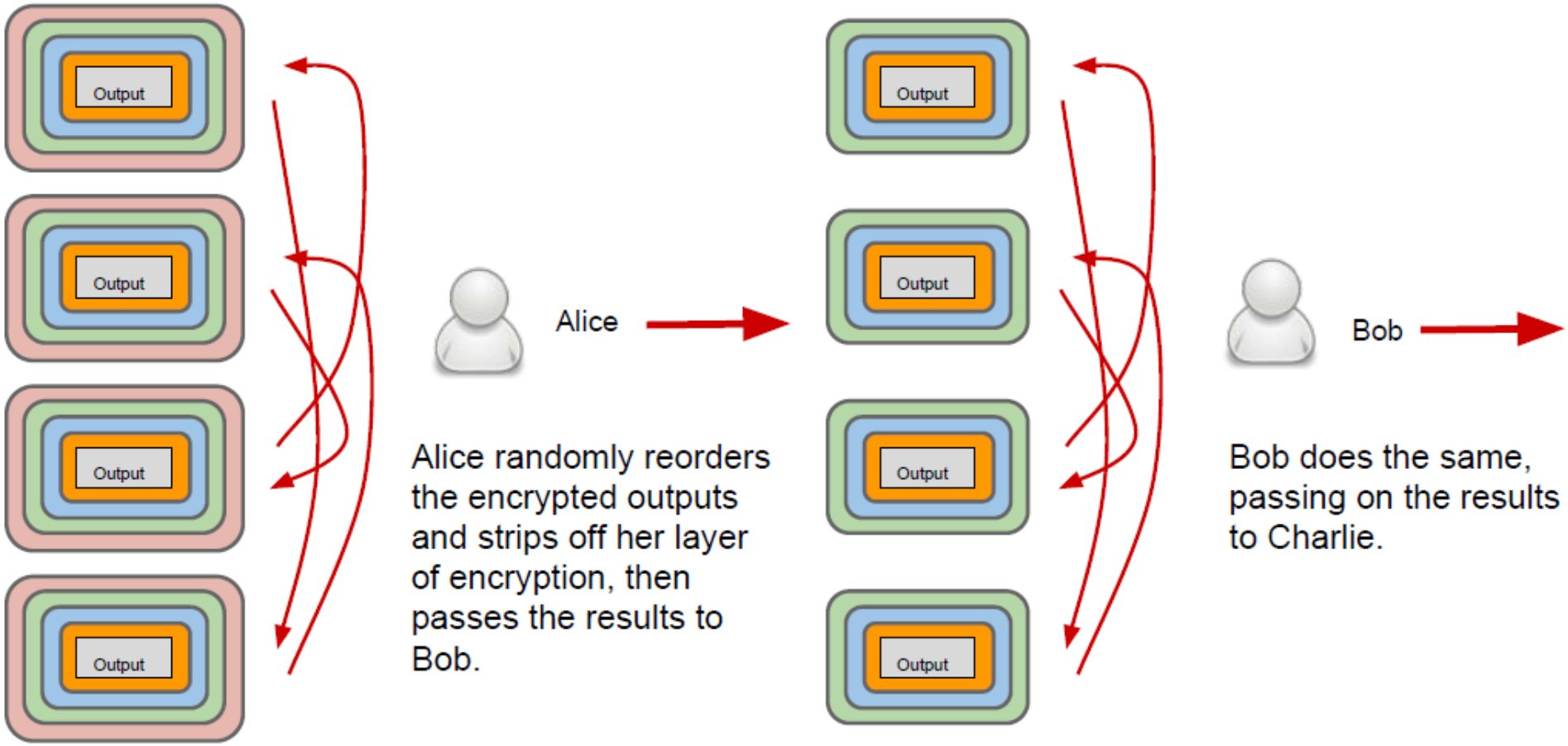


Step 2: Encrypt outputs

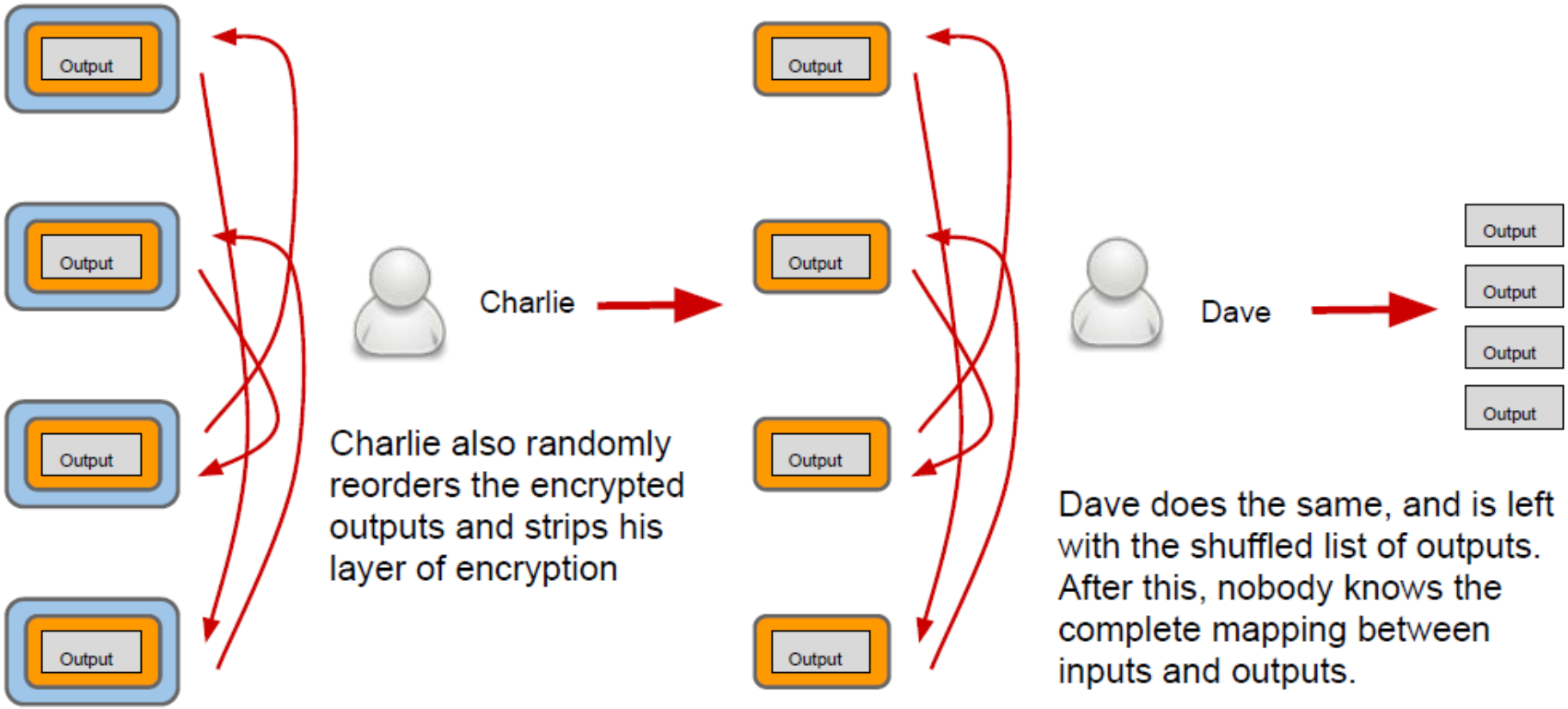




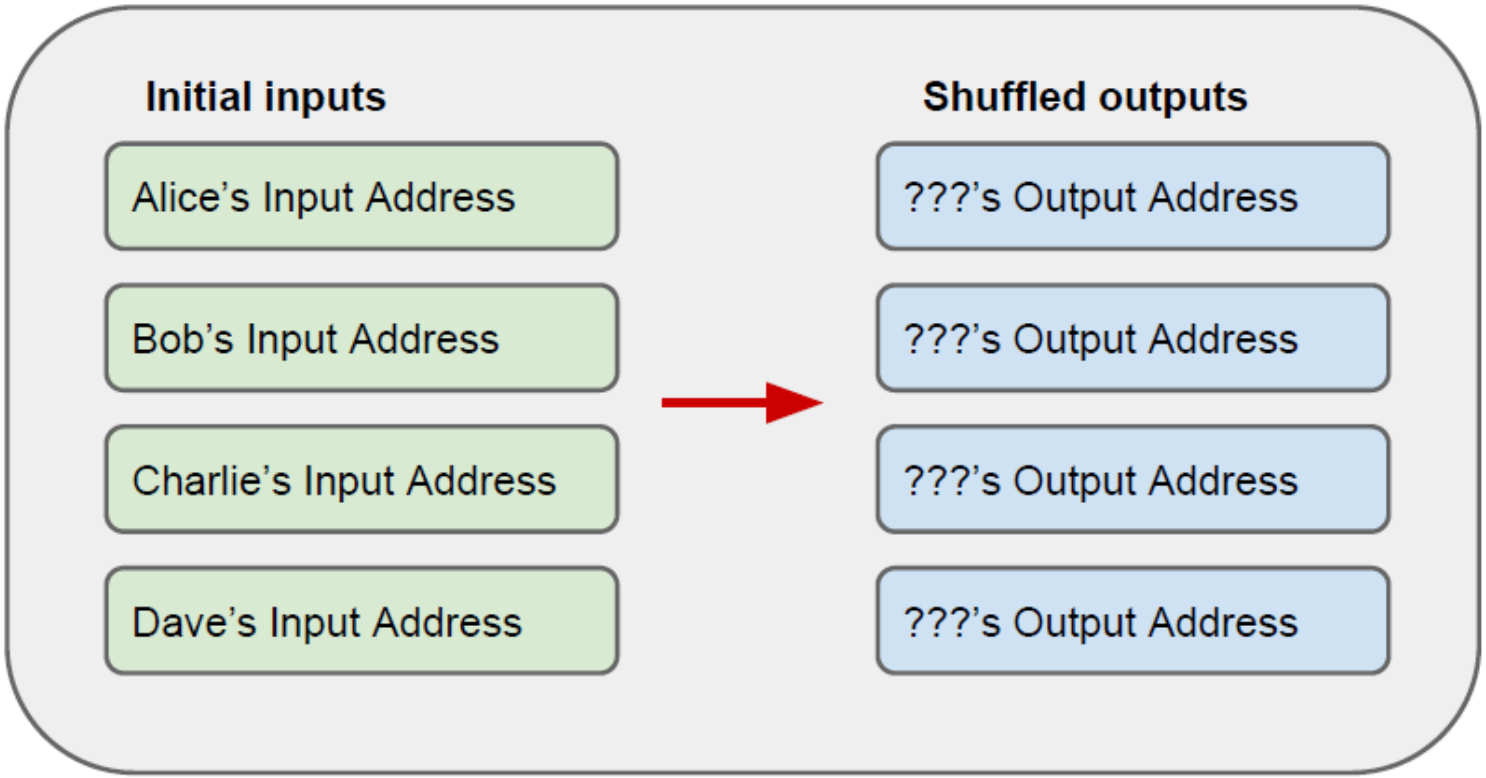
Step 3: Shuffle and Decrypt



Step 3: Shuffle and Decrypt

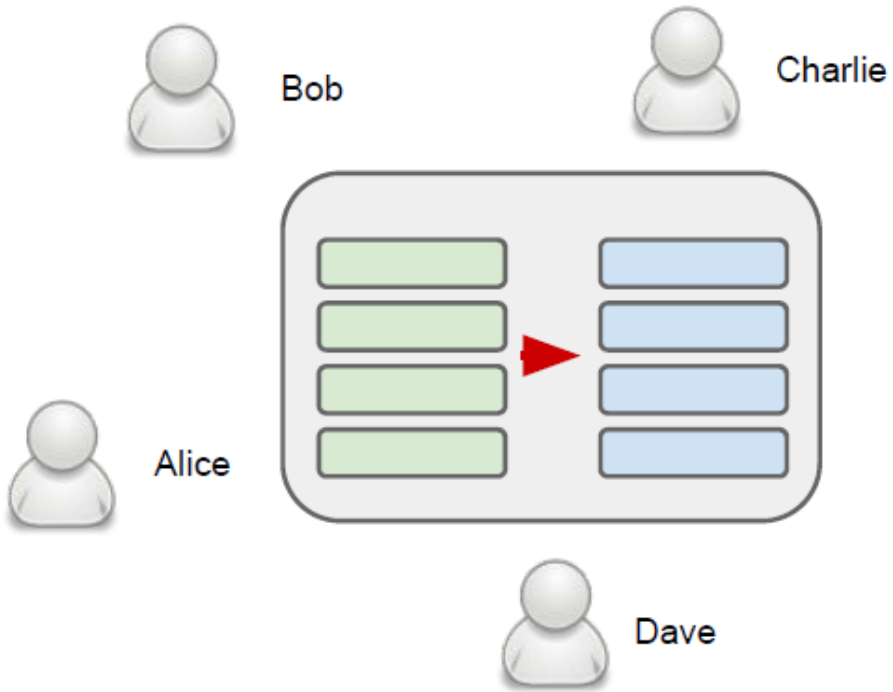


Step 4: Generate Unsigned CoinJoin



Step 5: Review, Sign, Broadcast

- Each participant reviews the transaction and verifies that his or her inputs and outputs are as expected.
- Each participant signs the transaction if it matches, then the signed transaction is broadcast.
- If transaction is claimed incorrect, enter Blame phase.



Other notes

- Assume there are 10 participants: 3 of them are honest, including you, and 7 are malicious and collude
- Assume the protocol has finished successfully

Other notes

- Assume there are 10 participants: 3 of them are honest, including you, and 7 are malicious and collude
- Assume the protocol has finished successfully
- Can the 7 malicious guys find out which of the 10 output addresses is yours?

Other notes

- The 7 malicious guys always know that your address is not among their 7 addresses, just because they know their own 7 addresses
- Thus your address can only be among the 3 remaining addresses

Other notes

- The 7 malicious guys always know that your address is not among their 7 addresses, just because they know their own 7 addresses
- Thus your address can only be among the 3 remaining addresses
- This “attack” is always possible in every form of coin mixing, no matter how you organize the mixing

Comparison

- Mixers – trusted party can steal, link inputs and outputs
- CoinJoin – trusted party can link inputs and outputs, cannot steal
- CoinShuffle – no trusted party