Blockchains & Distributed Ledgers

Lecture 08

Aggelos Kiayias

Slide credits: AK

Lecture Overview

- Anonymity & Privacy in blockchain protocols.
  - Bitcoin and CoinJoin transactions.
  - Mix-nets
  - group and ring signatures.
    - Cryptonote/Monero
  - Zero-knowledge proofs & SNARKs
    - Zcash.

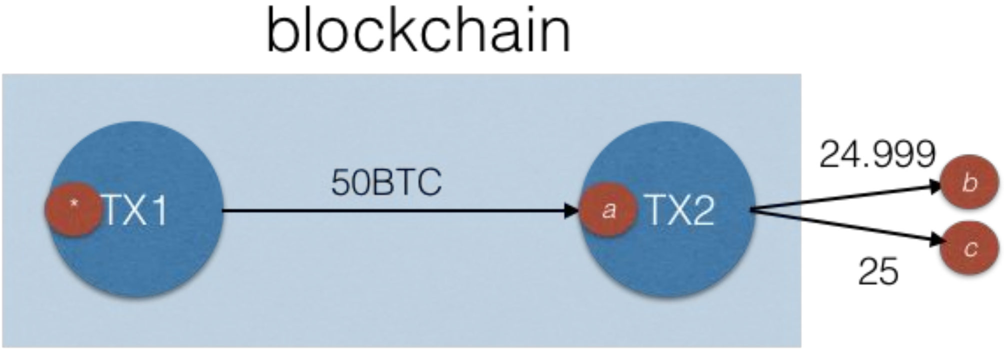
Pseudonymity vs. Anonymity

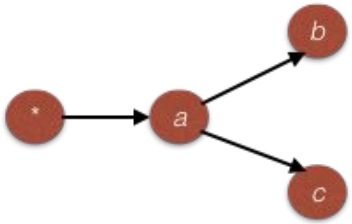
- Pseudonymity: identities are substituted by tags that are independently assigned to each identity.
- Anonymity: any action performed is manifested within a set of indistinguishably acting participants. (The anonymity set)

Privacy and Bitcoin

- Users can create accounts -practically- without cost and without association to previous accounts.
- Essentially they can create an unlimited number of pseudonyms.

Transaction Graph Analysis

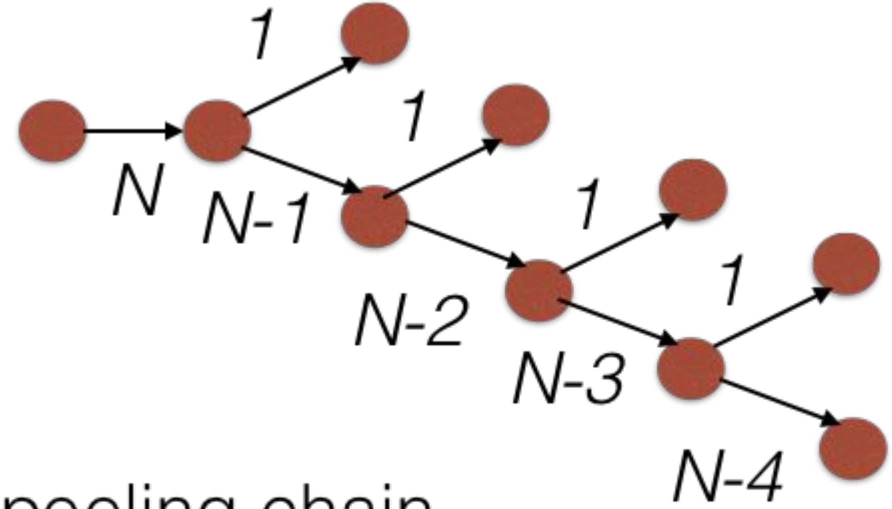




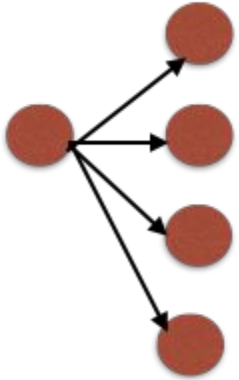
bitcoin
generation
transaction

account *a*
moves 50 BTC to
accounts *b* and *c*
(minus fees)

Common Behaviours



peeling chain

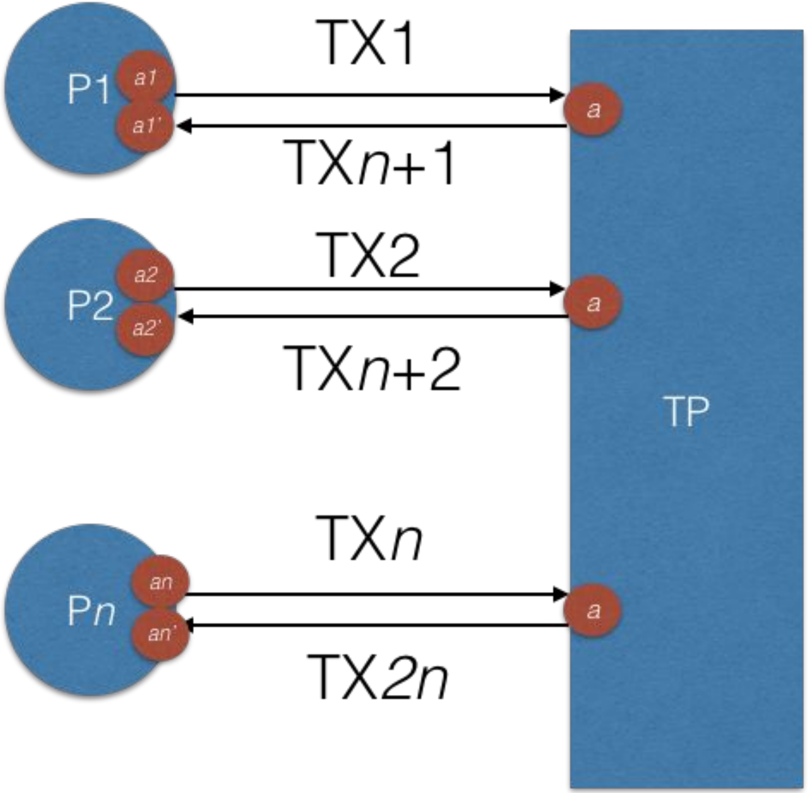


star

Fungibility and Privacy

- Coins are interchangeable.
- Since each “satoshi” has its whole history in the bitcoin blockchain, its fungibility is debatable.

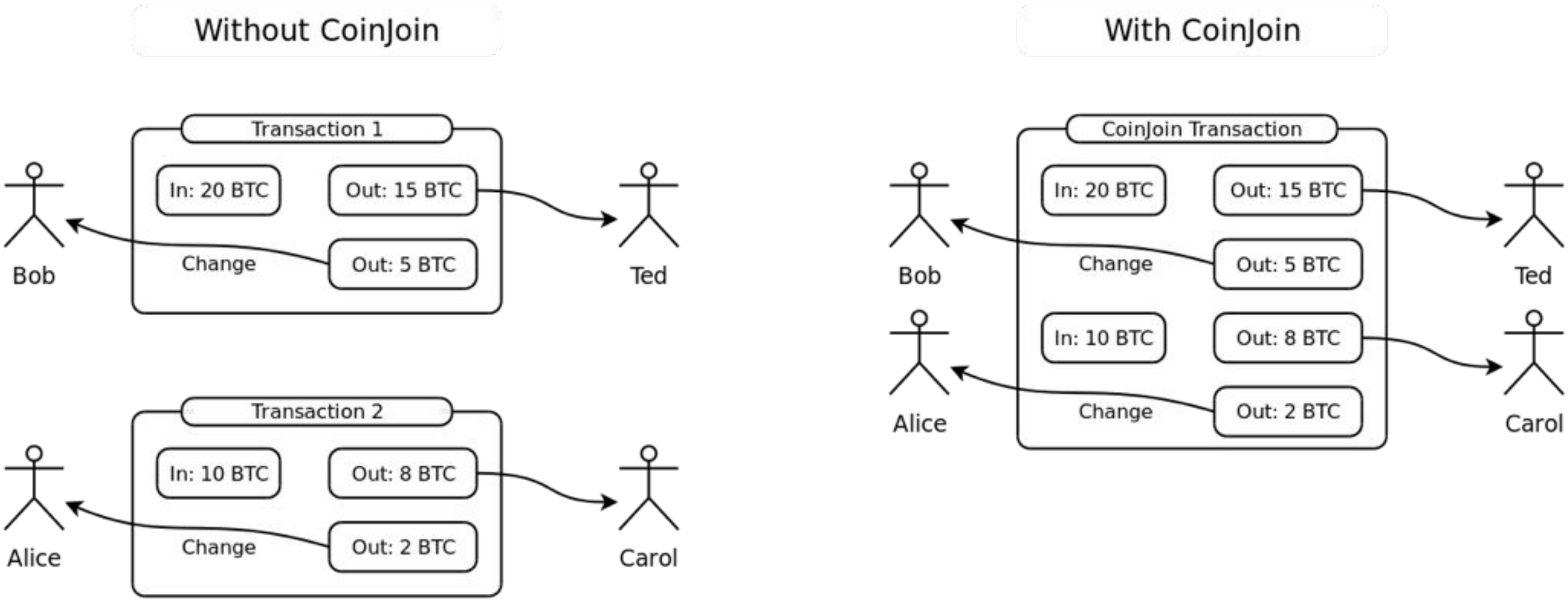
Anonymising Transactions



Anonymity set of size *n*

TP may disappear with the money

CoinJoin


Without CoinJoin
Transaction 1
In: 20 BTC
Out: 15 BTC
Out: 5 BTC
Change
Bob
Ted
Transaction 2
In: 10 BTC
Out: 8 BTC
Out: 2 BTC
Change
Alice
Carol
With CoinJoin
CoinJoin Transaction
In: 20 BTC
Out: 15 BTC
Out: 5 BTC
Change
In: 10 BTC
Out: 8 BTC
Out: 2 BTC
Change
Bob
Ted
Alice
Carol

Using Multiple Input Transactions

- *n* parties in communication. One party designated as leader.
- The *i*-th party sends to the leader, the recipient address b_i and return address c_i together with corresponding amounts. When all *n* parties complete this step, then the multiple input transaction can be formed by the leader and send to all *n* parties.
- parties send their signature to the leader. When all *n* signatures are received then the multiple input transaction can be posted on the blockchain by the leader.
- If any of the *n* parties aborts the protocol the transaction cannot be validated.
- **Questions**: is it possible to ensure that the adversary cannot do a correlation between a_i and b_i, c_i ? in case of an abort is it possible to restart the protocol without the offending party?

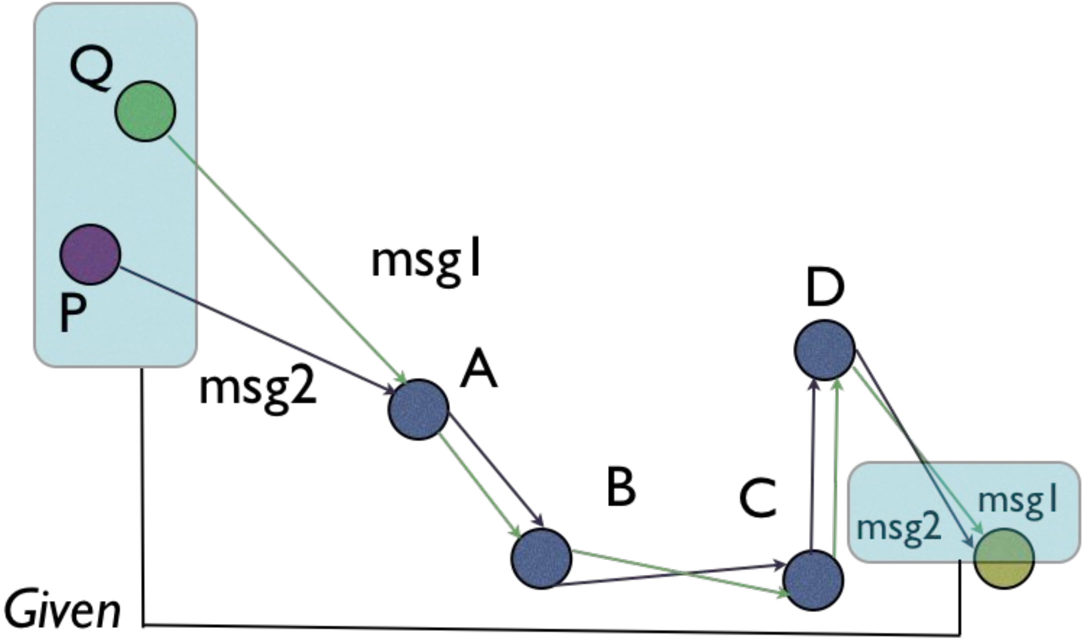
Passive vs Active Attacks

- A “passive” adversary would just observe the transaction in the blockchain. In this case, an anonymity set of size *n* may be considered for each participant assuming equal amounts.
- However, an “active" adversary participates in a protocol execution; the correlation between participants will be apparent to the leader in the multiple input transaction (even if communication with the leader is performed via an encrypted channel).

Mix-net

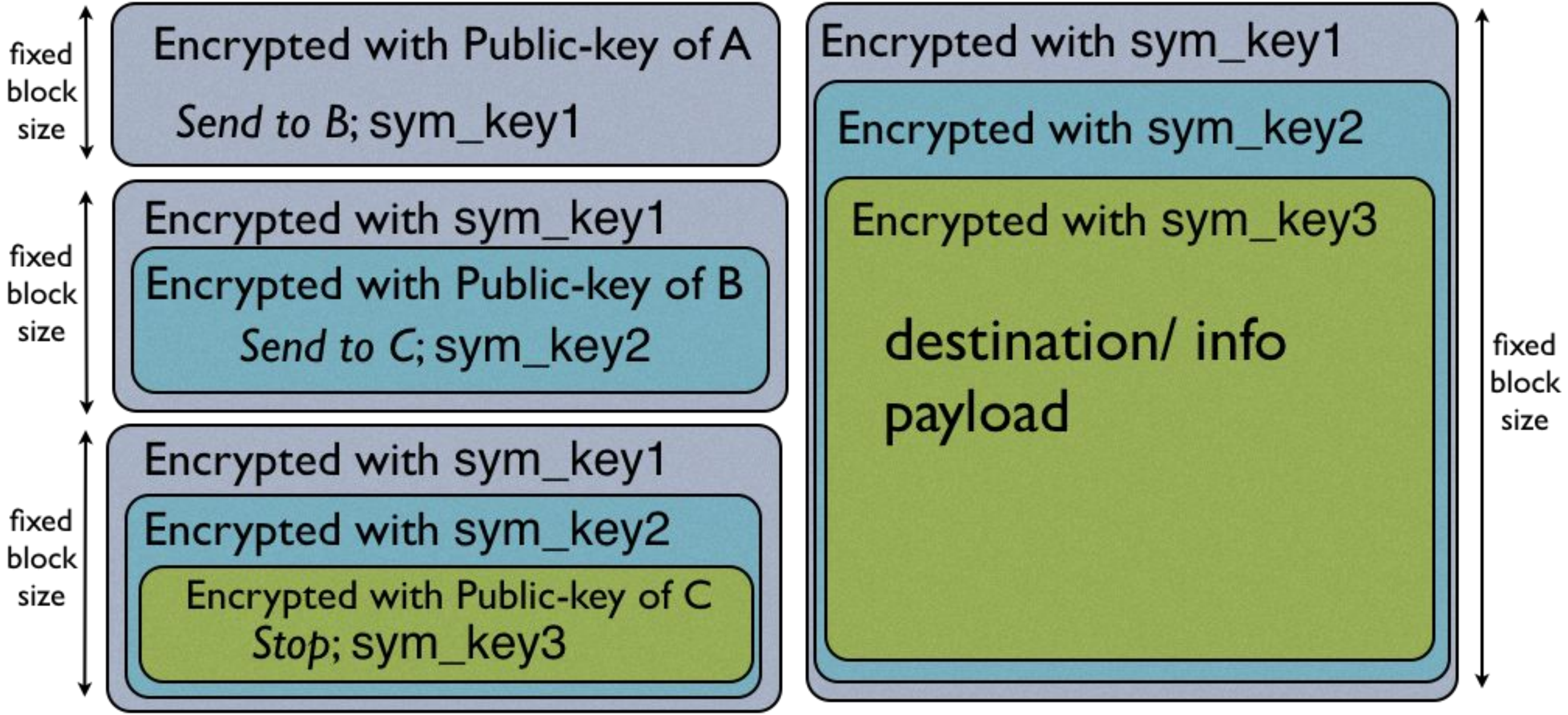
- Facilitates a sender-anonymous broadcast.
  - Decryption mix-nets.
  - Re-encryption mix-nets.

Mix-net, 2

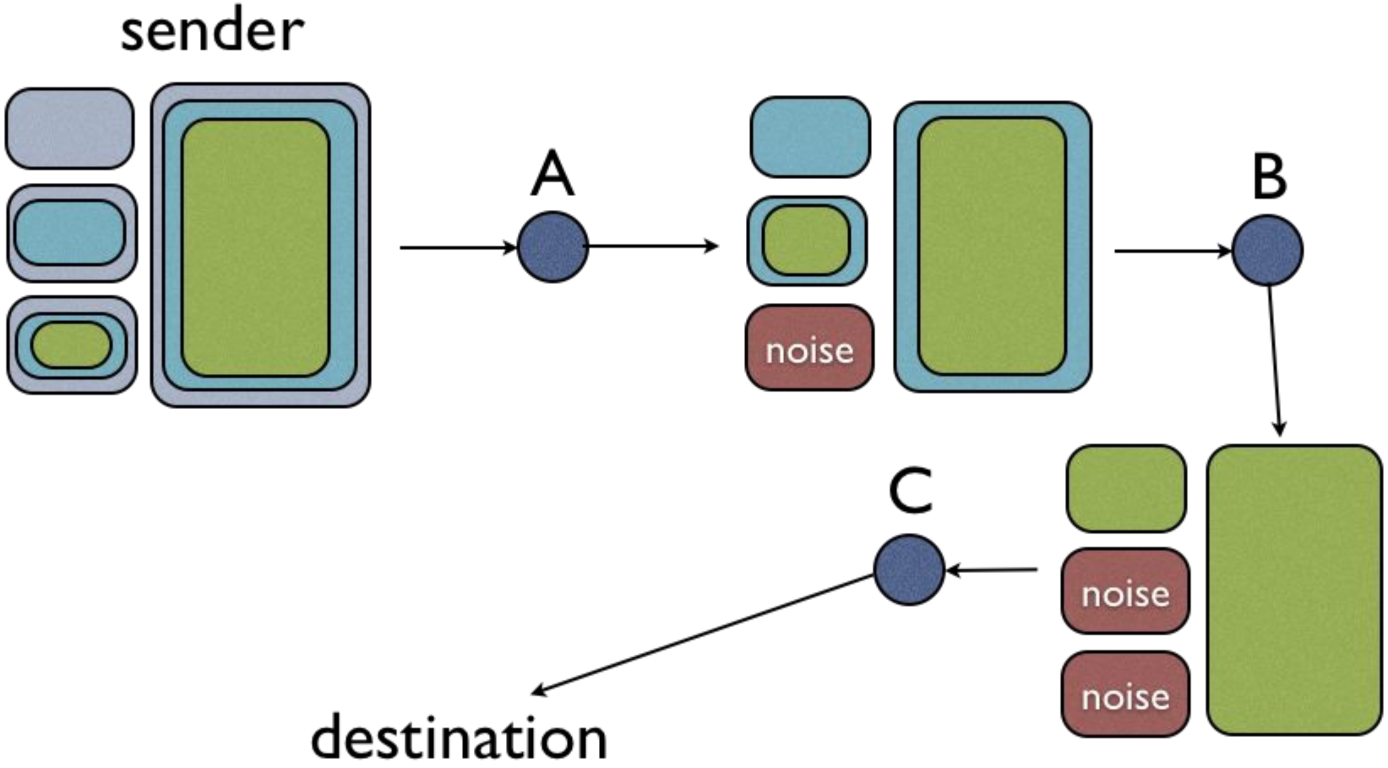


Not possible to relate whether P send msg1 or msg2 and similarly for Q (as long as there is one honest mix)

Decryption Mix-net



Routing via a Mix-net

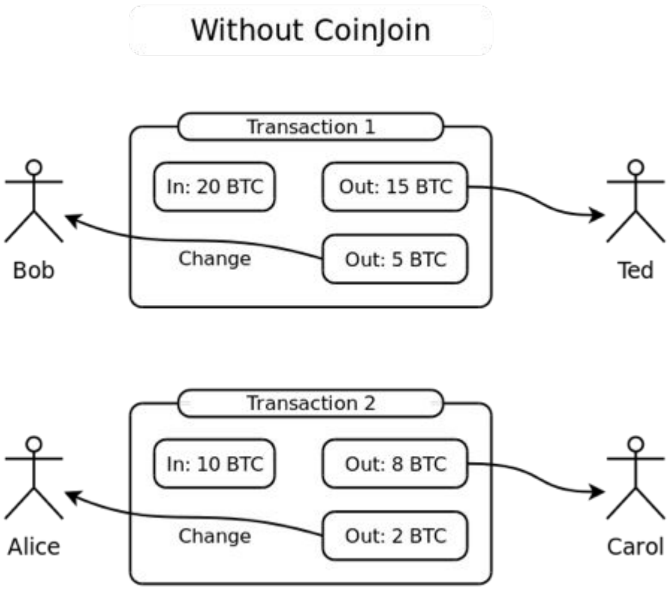


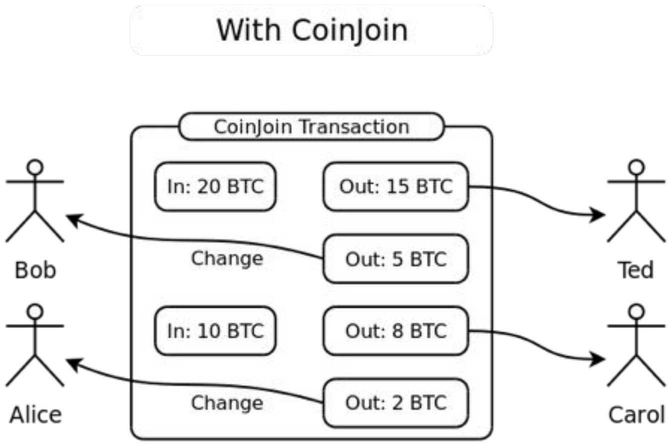
Mix-net for Coinjoin transactions

- Parties share with all parties their public-keys (the association between public-keys and accounts a1,a2, …, *an* is public).
- Parties engage in a decryption mix-net in sequence so that the last party is the leader and obtains all the relevant information to assemble the multiple input transaction which is then sent to all parties.
- Note that each step is performed by a designated party Pi, hence any abort can be attributed to that party. A repeat session may exclude the party Pi.
- Parties send their signatures to the leader.

Hiding Coin Balances: Mimblewimble

balances
are visible:



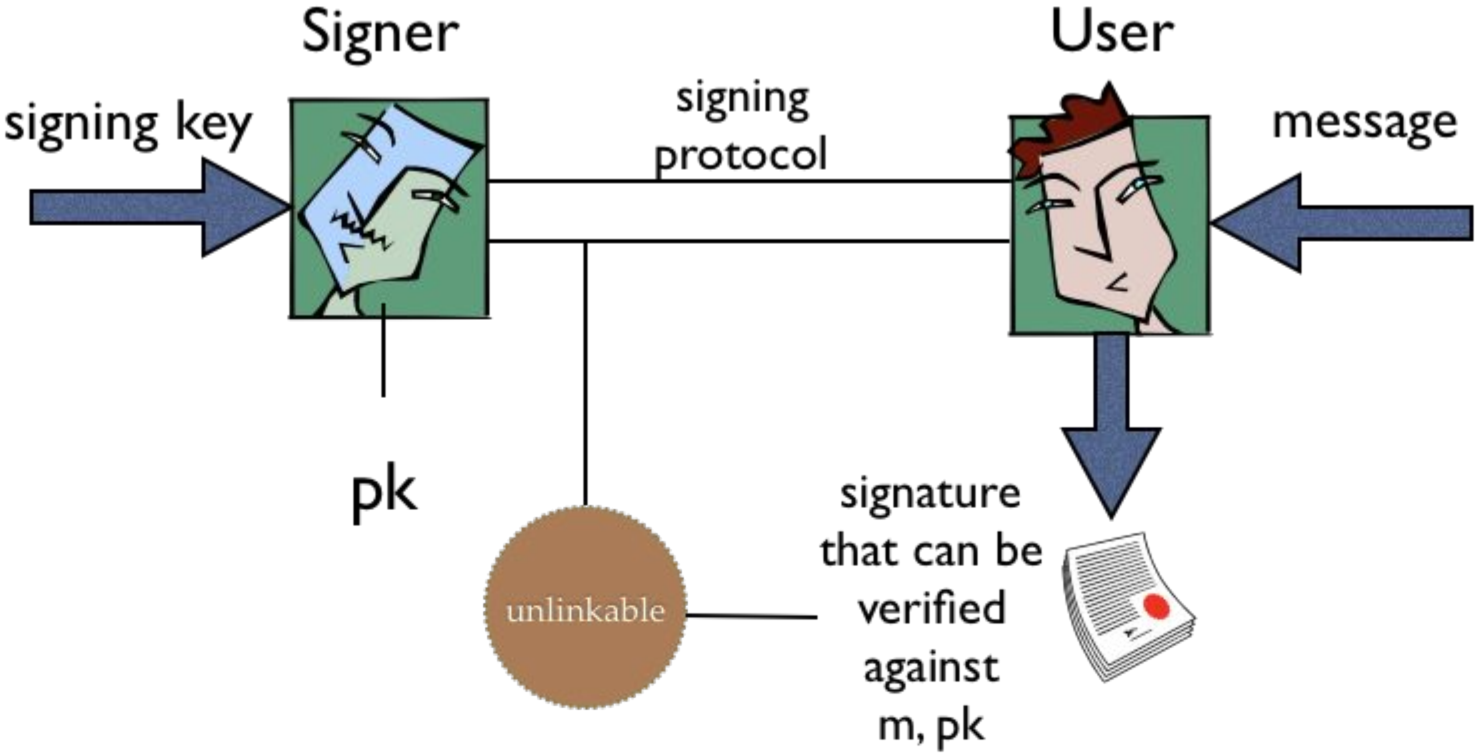


Mimblewimble: Use *commitments* with a homomorphic property:
Com(x) * Com(y) = Com(x+y)
Instead of revealing the balance transfered, commit to it
And ensure value preservation value via the homomorphic property

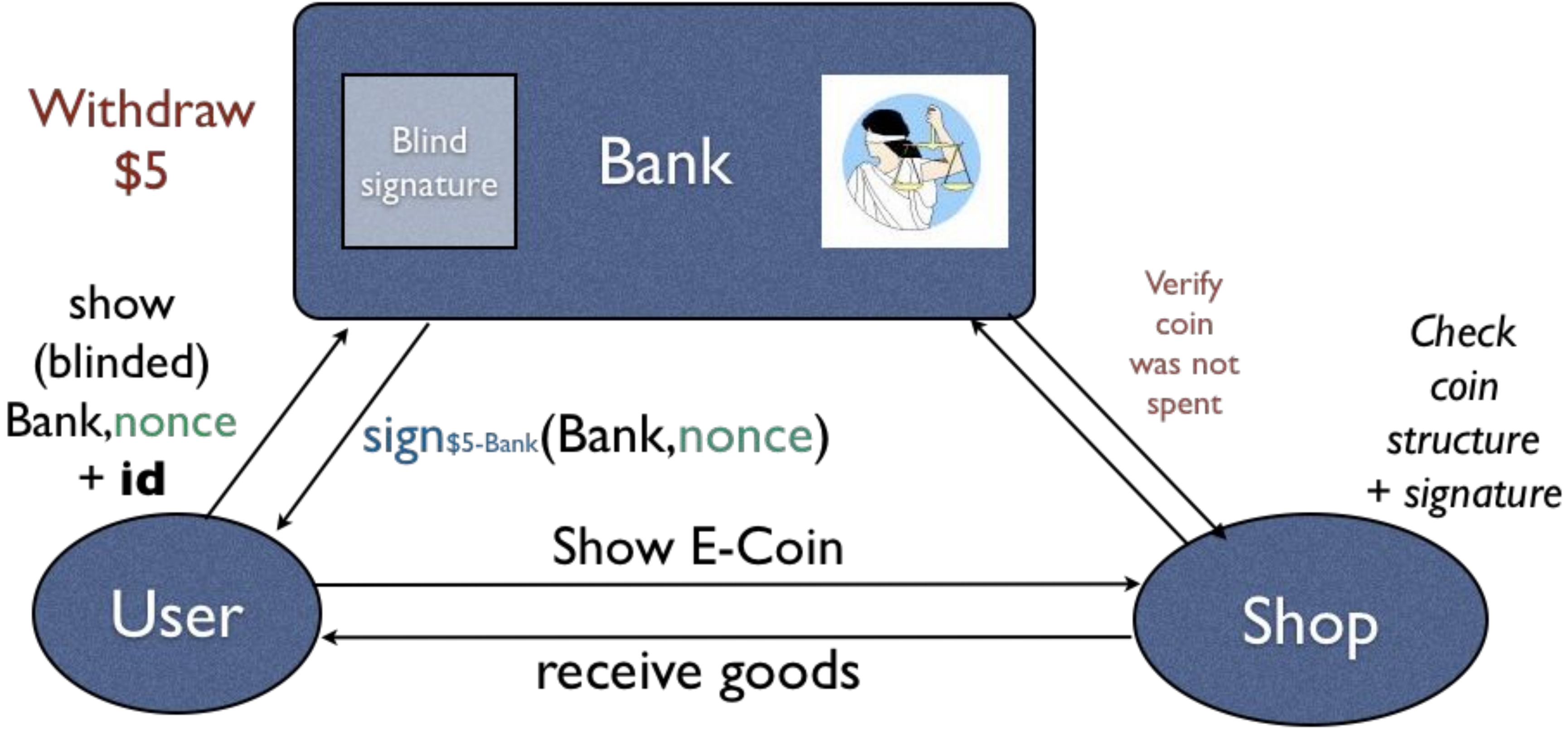
Coordination

- Coinjoin & similar techniques require coordination and message passing between multiple parties.
    a. How do parties find each other?
    b. How to deal with denial-of-service attacks?
- Is it possible to improve on these issues using more advanced cryptographic techniques?

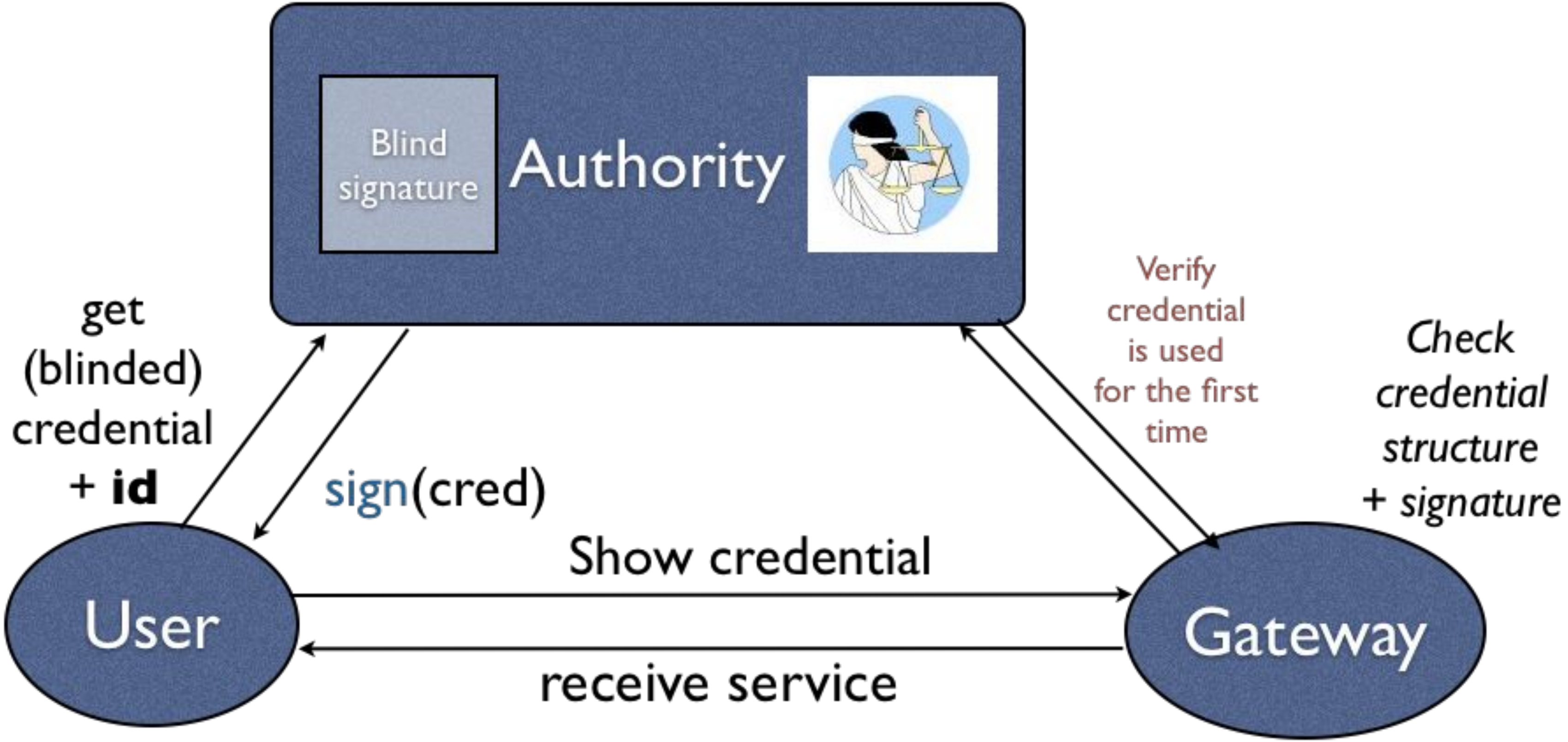
Blind-Signatures



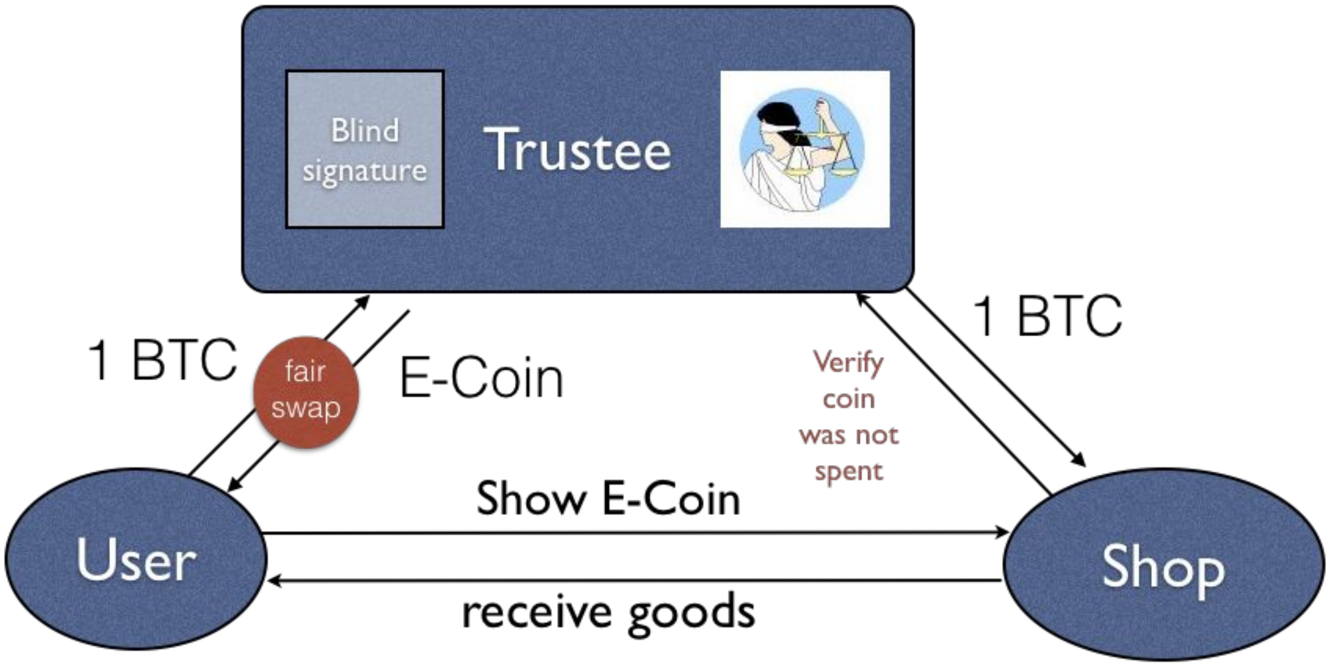
Chaum's E-cash



Anonymous Credentials



Anonymizing Bitcoin Payments via E-cash



Note: Trustee is trusted to honor its e-coins.

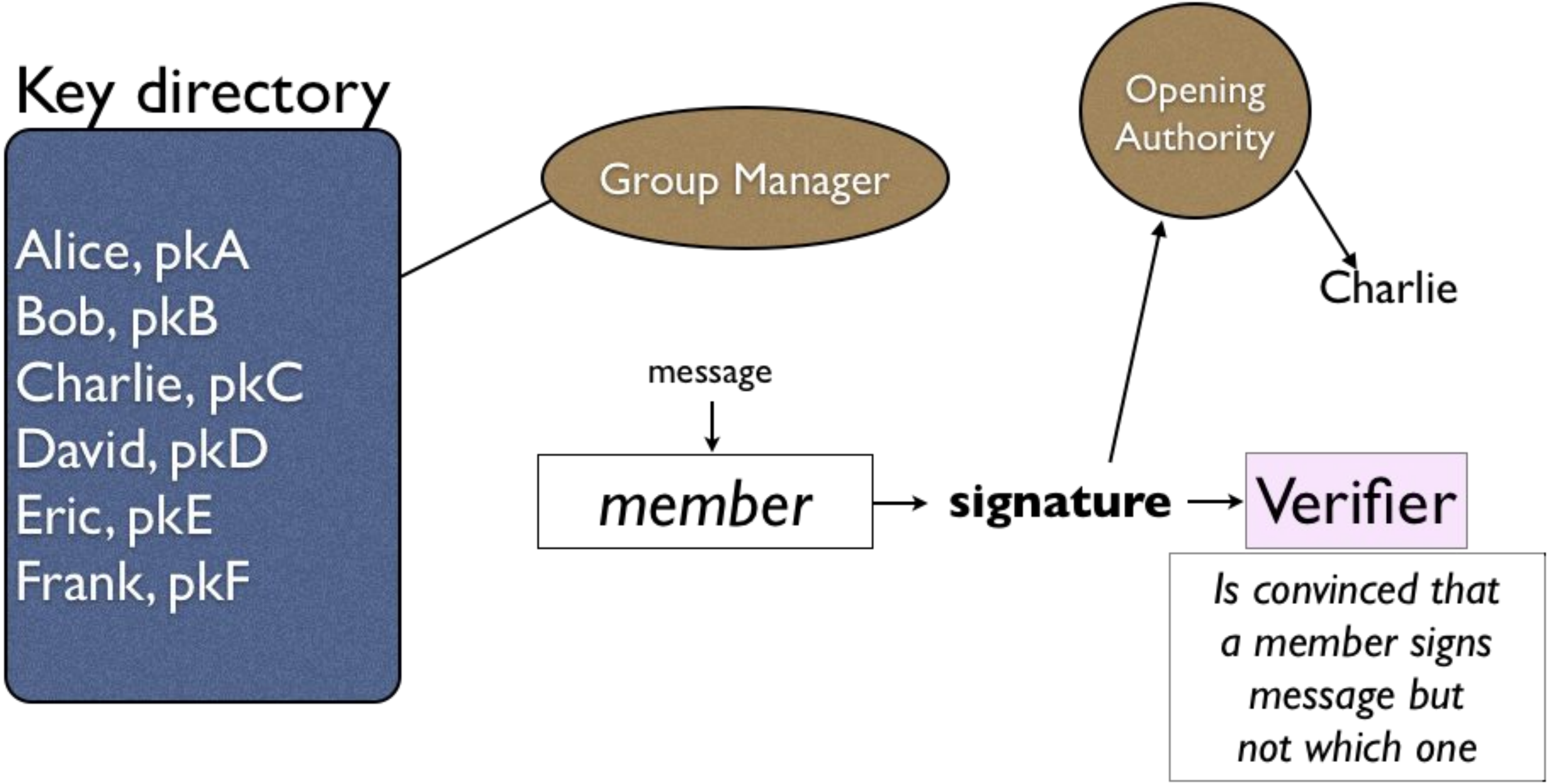
Fair Swaps

- Alice and Bob would like to exchange secrets so that either none of them gets their output or both.
  - Classical problem!
  - Impossible to solve under standard network assumptions!
  - Going around the impossibility: (i) optimistic fair exchange (ii) resource-based fair exchange (iii) fair swaps with penalties.
- [Construction] using a smart contract that both parties fund to accept their secrets. Key requirements: (i) parties lock up funds, (ii) secret submission should be verifiable by the contract code

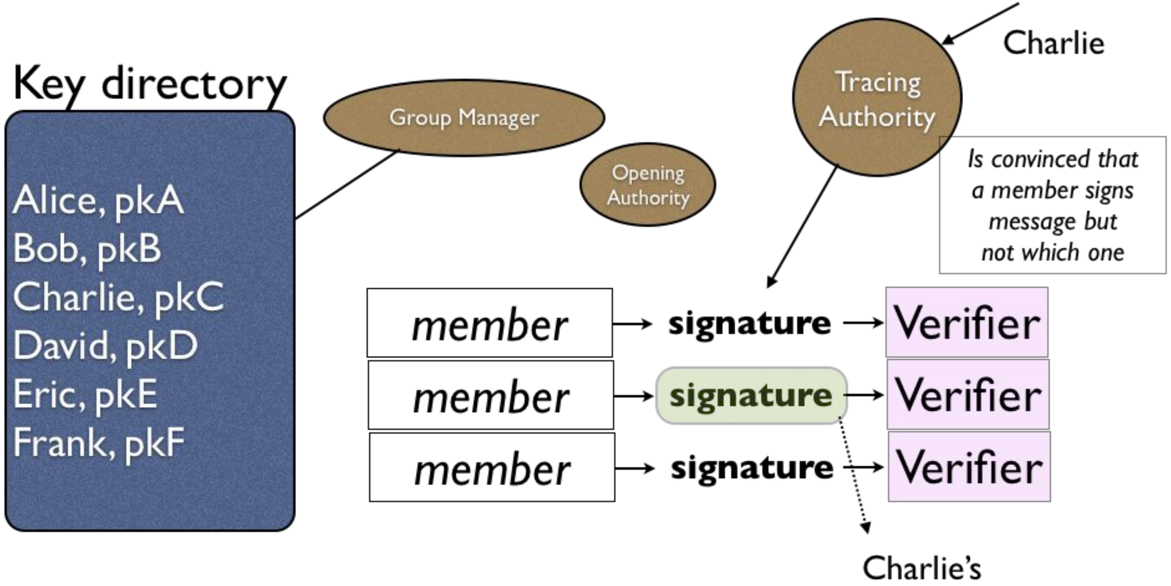
Anonymity and Digital Signatures

- So far all digital signatures identify the signer.
  - Is it possible to hide the sender within a group?

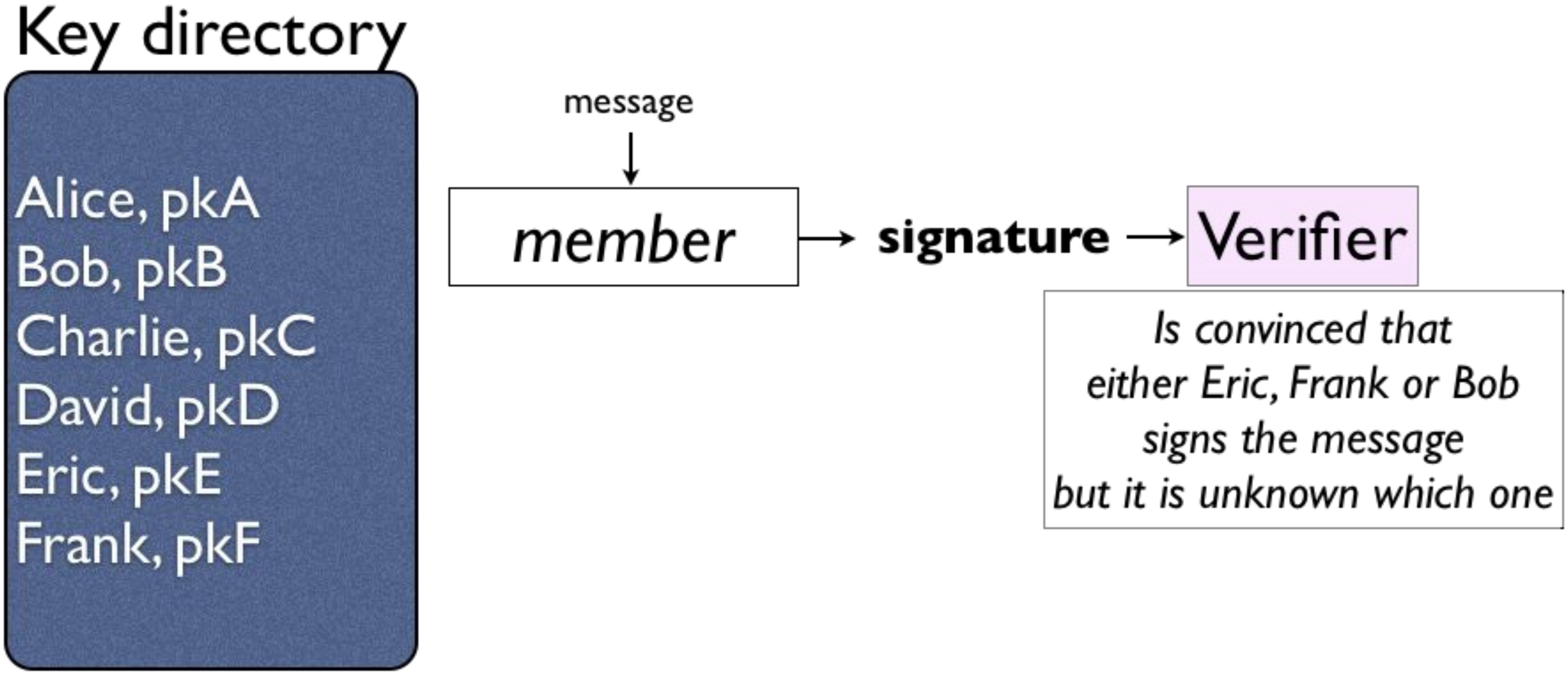
Group Signatures



Traceable Signatures



Ring Signatures



Monero/Cryptonote

- Uses *"stealth" addresses* and *linkable ring signatures* to provide better anonymity.
    a. For each payment an anonymity set is selected with accounts of the same monetary value.
    b. A ring signature is issued on behalf of that set, suitably restricted so that an account can only be used once (if twice it is *linkable*).
    c. Stealth addresses enable the sender to create unlinkable addresses for the receiver which subsequently the receiver can detect.

Is Monero Anonymous?

- There is potentially more uncertainty in the Monero blockchain compared to a bitcoin-like blockchain (even with Coinjoin transactions).
- Nevertheless, it is not obvious how to quantify the level of anonymization.
- De-anonymization is feasible in reasonable real world threat models. (e.g., imagine the attacker “spraying” the ledger with transactions so that it commands a good number of selected accounts).

Increasing and Safeguarding the anonymity set, I

- A larger anonymity set is most preferable.
- However in the techniques we have seen so far, transaction preparation work increases linearly with the anonymity set.
- **Ideal:** use the set of all possible unspent transaction outputs.

Increasing the anonymity set, II

Back to commitments!

$$\langle \rho, sn, \psi = \underbrace{\text{Commit}(\rho, sn)}_{\text{public}} \rangle$$

The commitment value is associated with a deposit to the ledger ("minting" a coin for $1).

Spending a coin, requires announcing the *sn* and proving that it was committed before in the ledger; (withdrawing $1)

sn

Is a valid $1 coin

$$\underbrace{\exists i : \psi_i}_{} = \text{Commit}(\rho, sn)$$

existential quantifier over all commitments in the blockchain

Increasing the anonymity set, III

Organize all commitments and serial numbers in a Merkle tree.

Prove that there is a leaf in the Merkle tree that contains the commitment

$$\psi_i = \mathrm{Commit}(\rho, sn)$$

Statement representation and witness size logarithmic in the number of coins.

Challenges

- How is it possible to prove efficiently statement referring to the leaf of a Merkle tree?
  - a possible solution: use “ZK-snarks”
  - SNARK = succinct non-interactive argument of knowledge.
- How to transfer a coin from one user to another? (one cannot simply transfer ρ).

ZK-Snarks

- Zero-knowledge succinct arguments of knowledge.
    - Similar to “zero-knowledge proofs”
        - Can prove possession of a witness for any public statement / predicate.
        - *With computational* soundness: it depends on the security of a “common reference string”; this is structured cryptographic information that is assumed to be honestly sampled.
        - With succinctness: the proof size and the verifier’s running time is efficient - proportional to the *statement* only.

Constructing ZK-SNARKs

- There exist a SNARK for any NP-relation R.

  NP = { L | exists R: x in L iff (x,w) in R; R is polynomial time}

- The actual proof sizes are small (hundreds of bytes)
- Verification does not depend on the running time of *R.*

Zerocash

$$\langle a_{\mathbf{pk}}, \underbrace{v}_{\text{value}}, \underbrace{s}_{\text{random}} \rangle$$

$(a_{\mathbf{pk}}, a_{\mathbf{sk}})$

account
public/secret
key

$$k = \mathrm{Commit}(\rho, a_{\mathsf{pk}}||s)$$

$$sn = \mathsf{PRF}^{\mathsf{sn}}_{a_{\mathbf{sk}}}(s)$$

$$\psi = \mathrm{Commit}(\rho', v||k)$$

coin : $\langle a_{\mathsf{pk}}, v, s, \rho, \rho', \psi \rangle$

PRF = pseudorandom function

Commit = commitment function

The double commitment enables verifying that the value *v* is properly encoded in the coin without revealing information about the owner

Zerocash "Pour" Operation

given coin $\langle a_{\mathsf{pk}}, v, s, \rho, \rho', \psi \rangle$

produce two new coins with values $v_1 + v_2 = v$

$a^1_{\mathsf{pk}}, a^2_{\mathsf{pk}}$

set $k_i = \mathrm{Commit}(\rho_i, a^i_{\mathsf{pk}} || s_i)$

$\psi_i = \mathrm{Commit}(\rho'_i, v_i || k_i)$

Reveal ψ_1, ψ_2 and prove that the Merkle tree has a commitment corresponding to a coin $\langle a_{\mathsf{pk}}, v, s, \rho, \rho', \psi \rangle$ that is split properly and a_{sk} is known

Serial number of spent coin is revealed and marked as spent

Include a public-key encryption of opening the commitment that the recipient can use to decrypt the coin secret values.

Common Reference Strings

- SNARKs require the exists of a “common reference string”
- A trusted computation is needed to produce it.
  a. Use *secure multiparty computation* (topic of our next lecture)
  b. Use *updateable reference strings* (URS) instead and outsource the update operation to miners/blockchain participants.
  c. Use *alternatives* to SNARKs that do not require it (disadvantage: performance would be worse