Blockchains & Distributed Ledgers

Lecture 08

Aggelos Kiayias

Slide credits: AK, Dimitris Karakostas

Eponymous system

- Each action can be **attributed** to a user's **real-world identity**
- Examples:
  - Facebook - posts/comments are linked with the real-world name of the user who made it
  - Tweeter blue check (pre-X) - accounts are verified w.r.t. real-world identification documents
  - UK parliament votes - the vote of each MP is (publicly) attributable to each

Pseudonymous system

- Identities are represented as **tags**
- Each tag is independently assigned to each identity
- An **identity** may be **assigned multiple tags** and vice versa
- Examples:
    - X/Reddit - posts/comments are linked to an (arbitrary) username
    - Email - each message is linked with an email address
    - Graffiti - each piece is signed by a tag/pseudonym (e.g., Banksy)
- Sybil Attack:
    - The setting where a single operator manages multiple identities/pseudonyms in a system

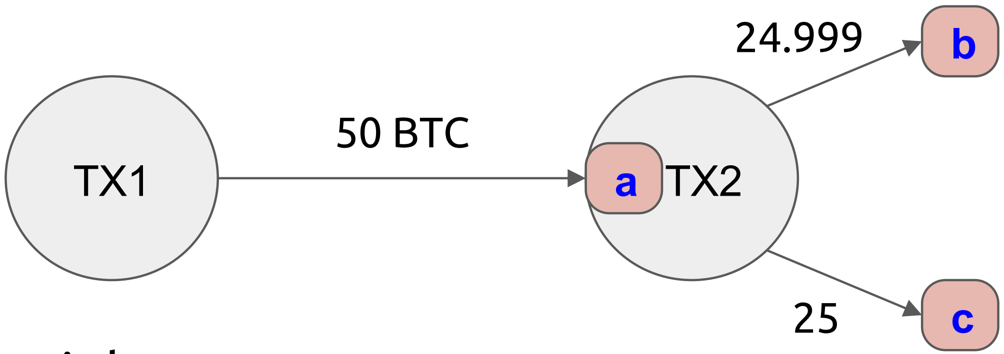
Anonymous system

- Any performed action is manifested within a set of **indistinguishably-acting participants**
- The set of indistinguishable participants is called **the anonymity set**
  - Hide in public
- Examples:
  - General election voting - e.g., 20,805 of 78,411 electorate voted Labour in 2024 Edinburgh North and Leith.
  - Tor browsing - website/hidden service sees only number of Tor connections (not name/IP)

Privacy in Bitcoin

- Users can create multiple accounts/addresses:
  - without cost
  - without association to previous accounts
- Essentially, users can create an **unlimited number of pseudonyms**

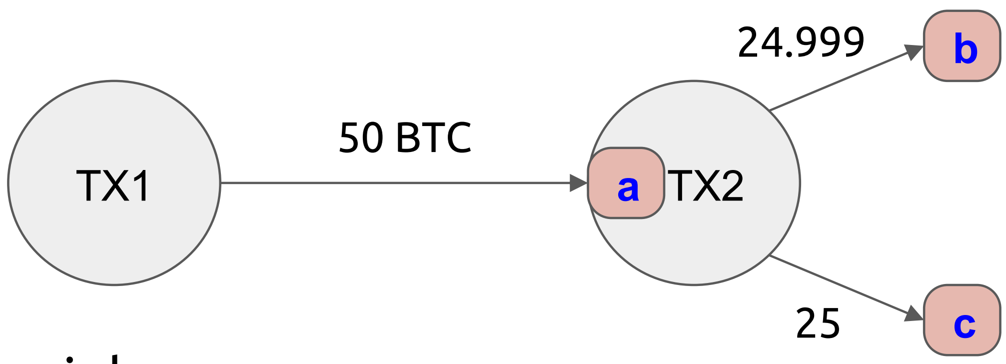
Transaction Graph Analysis



coinbase transaction

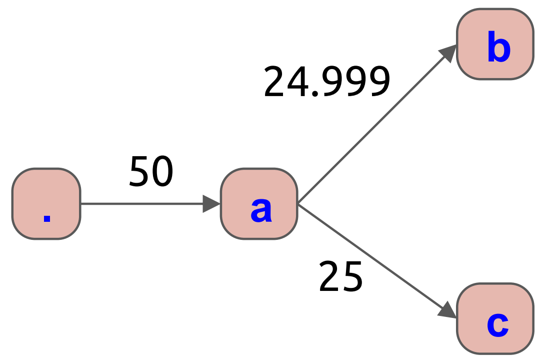
account **a** moves 50 BTC to accounts **b** and **c** (minus fees)

Transaction Graph Analysis

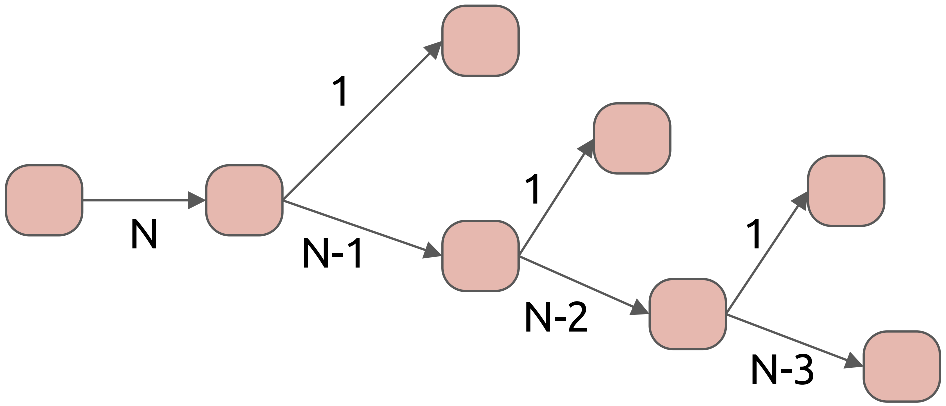


coinbase transaction

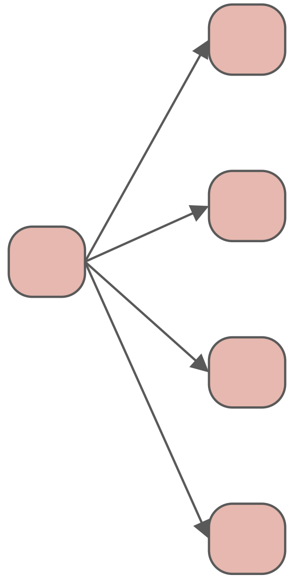
account **a** moves 50 BTC to accounts **b** and **c** (minus fees)



Common Behaviours



peeling chain

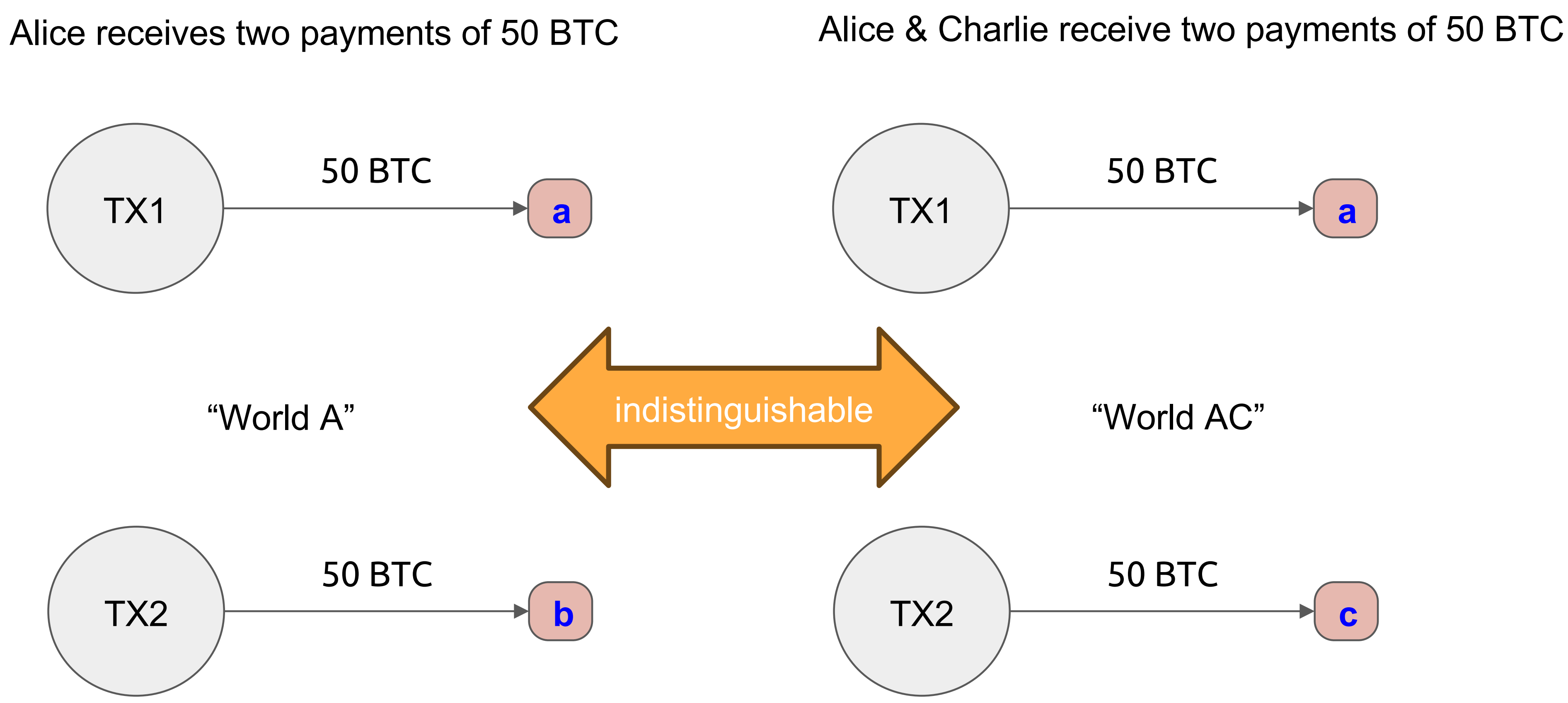


star

Fungibility and Privacy

- **Fungibility**: Coins are interchangeable
- However, each “satoshi” has its whole history in the Bitcoin blockchain
  - satoshi fungibility is debatable

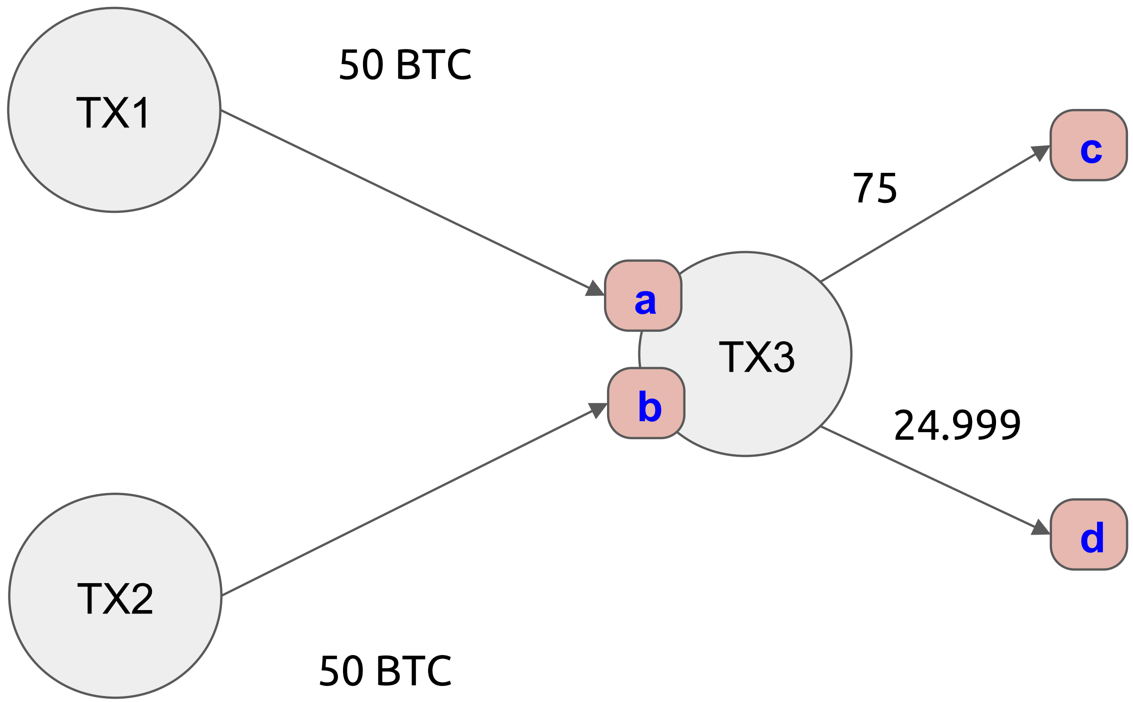
Bitcoin’s privacy



Two random variables are called (perfectly) indistinguishable if they produce any output with identical probability

Bitcoin's privacy (or lack thereof)

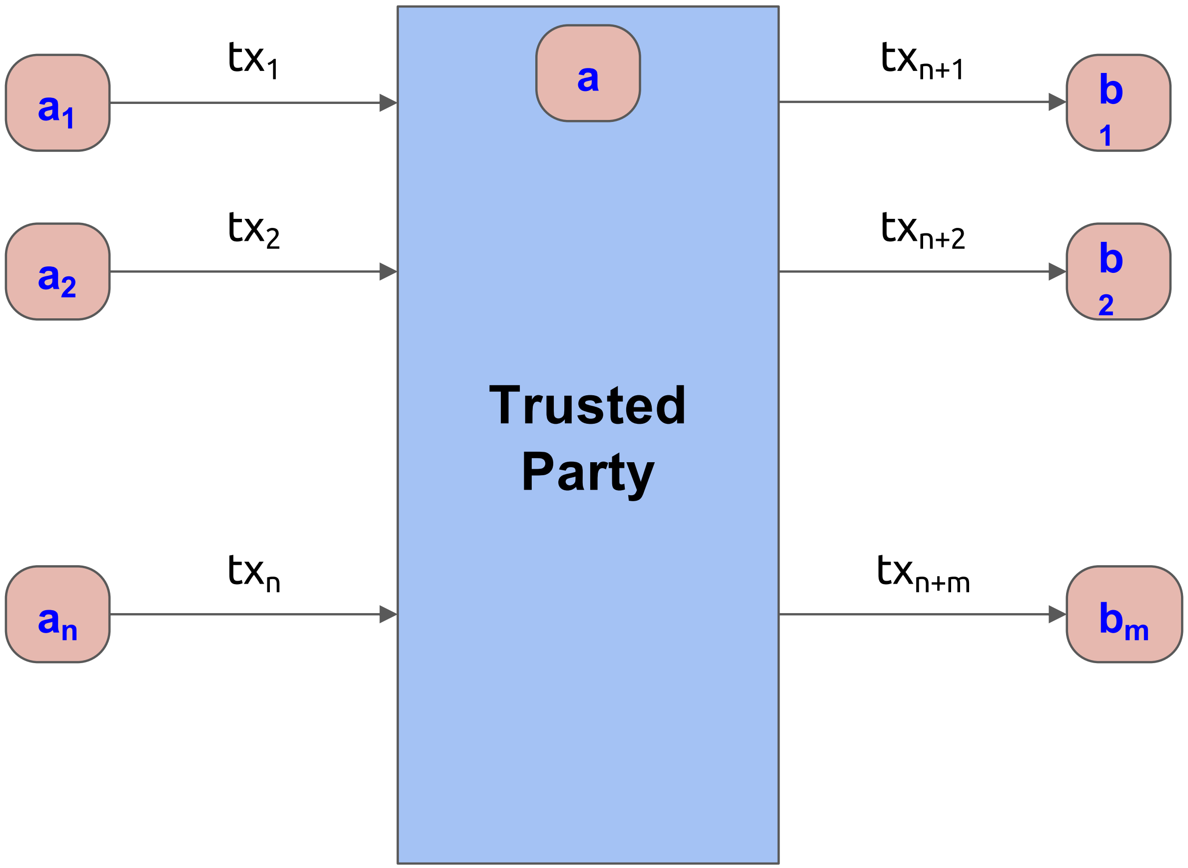
In world A, Alice wishes to make a payment of 75 BTC



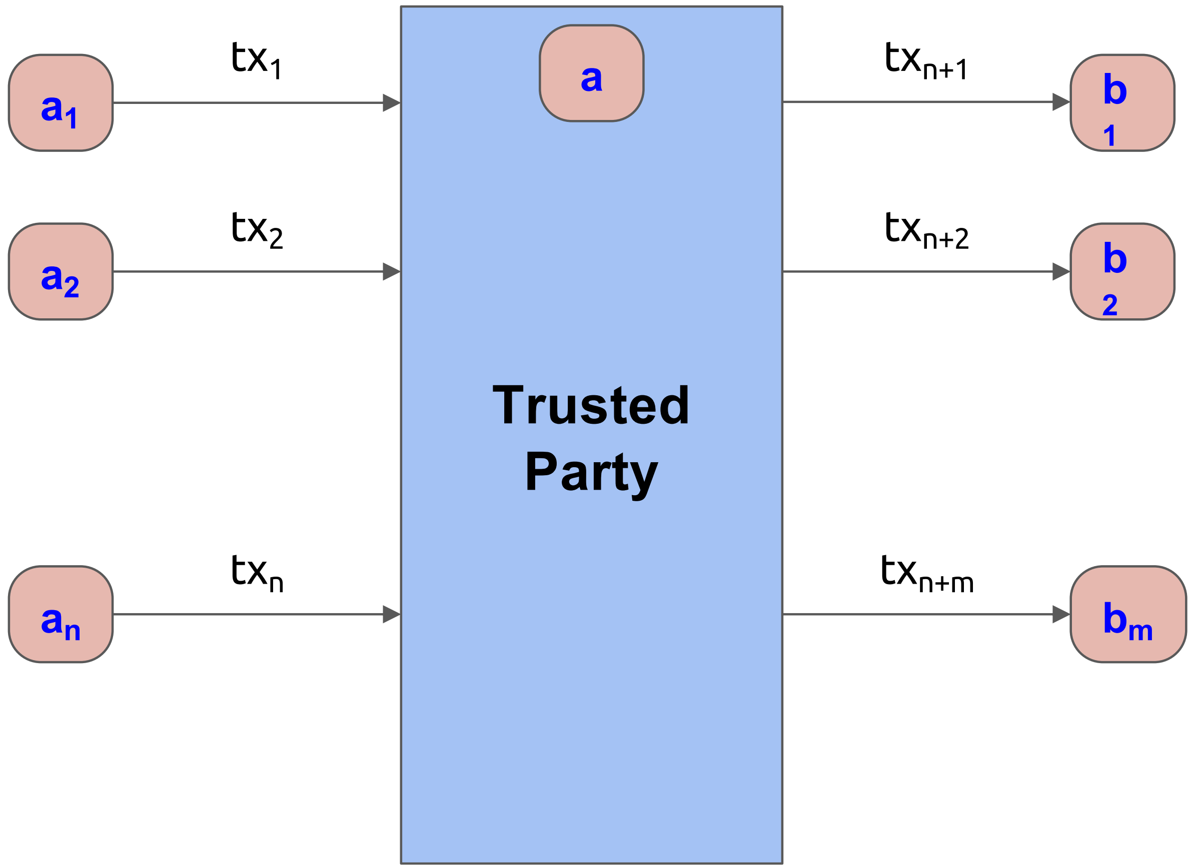
Given this observation,
an external third party can
conclude that it is
more likely to be in
"World A" instead of "World AC"

Transaction Anonymization Techniques

Anonymising Transactions - Centralized



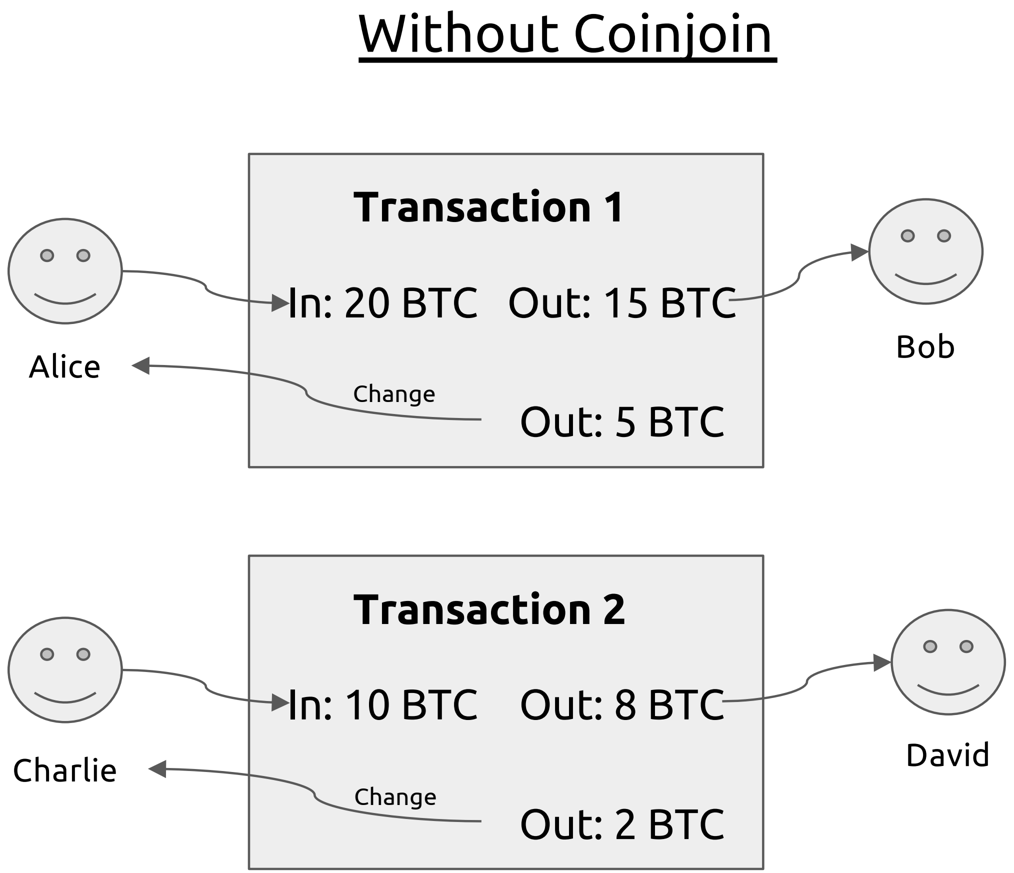
Anonymising Transactions - Centralized



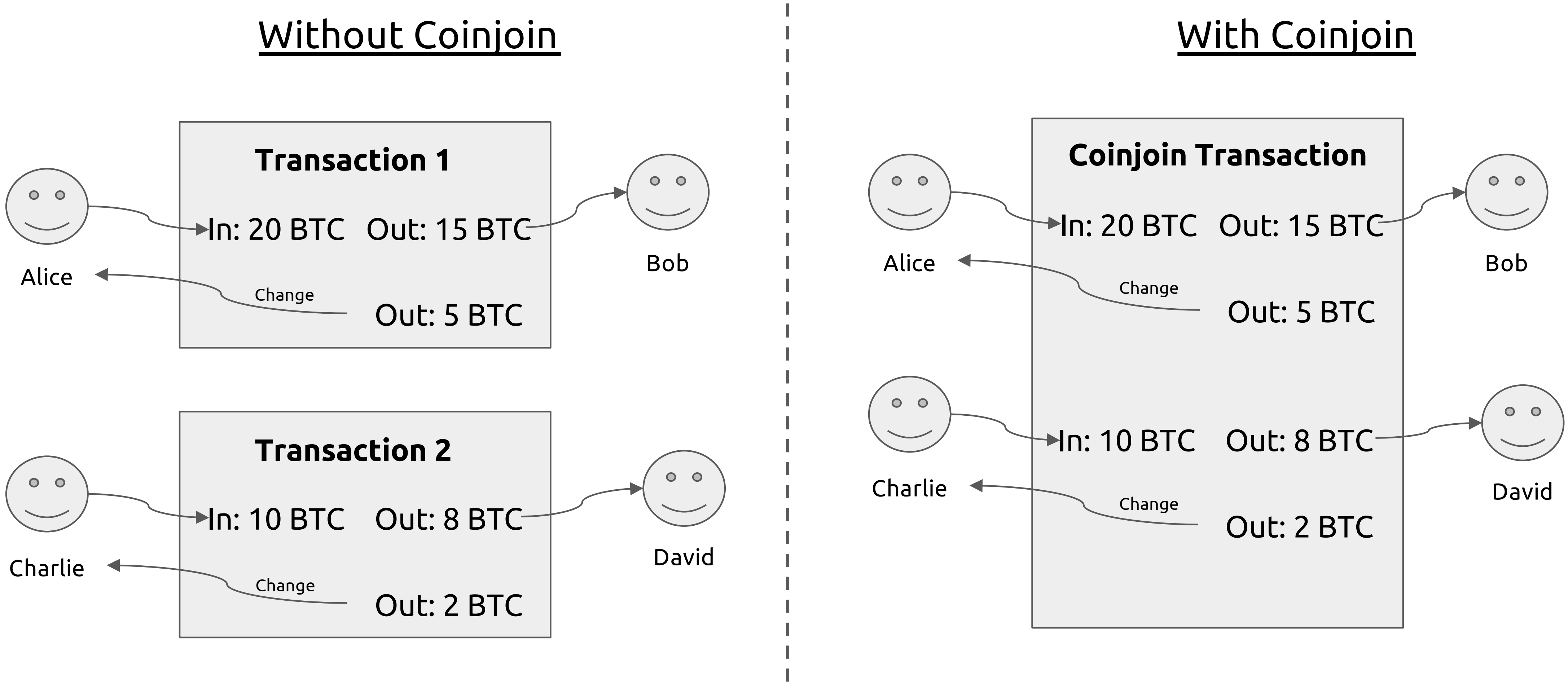
Anonymity set of size n

TP may disappear with the money

Anonymizing Transactions - CoinJoin



Anonymizing Transactions - CoinJoin



Multiple Input Transactions - Setup

- Parties:
  - *n* participants
  - one designated leader
- The *i*-th party sends to the leader:
  - the recipient address b_i
  - the return (change) address c_i
  - the corresponding amounts
- When all *n* parties complete this step, the multiple input transaction is formed by the leader and sent to all *n* parties

Multiple Input Transactions - Sign and Publish

- **Each party** sends a **signature** on the multiple input tx to the leader
- When all *n* signatures are received, the multiple input tx is **posted** on the blockchain **by the leader**
- If any of the *n* parties **aborts** the protocol, the transaction cannot be validated
- If the **leader is adversarial**, transaction cannot be published/validated
- Questions:
  - Can we ensure that an adversary does not correlate b_i, c_i ?
  - In case of an abort, is it possible to restart the protocol without the offending party?
  - Consider the following objectives; are they attained or not (and why)?
    - Denial of service protection
    - Loss of funds prevention
    - Privacy protection
    - Guaranteed termination

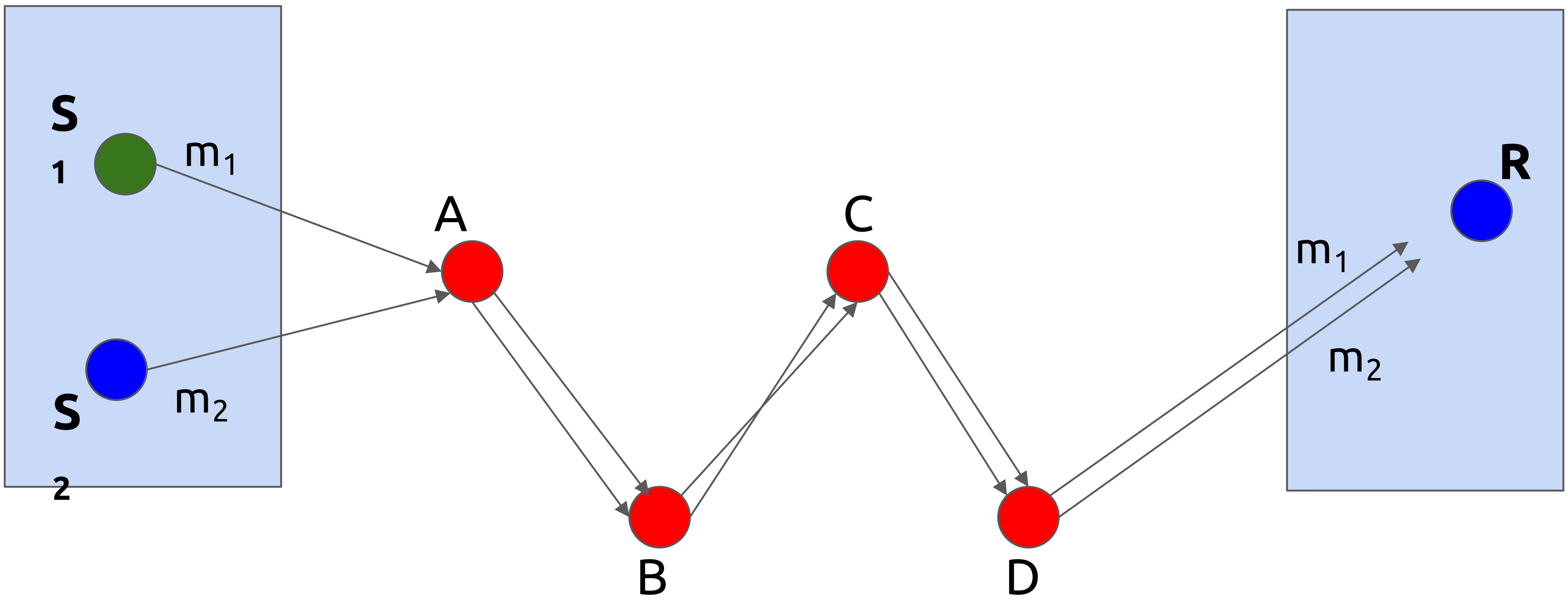
Passive vs Active Attacks

- A **passive** adversary just observes the network and the blockchain
  - an anonymity set of size *n* for each participant, assuming equal amounts
- 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

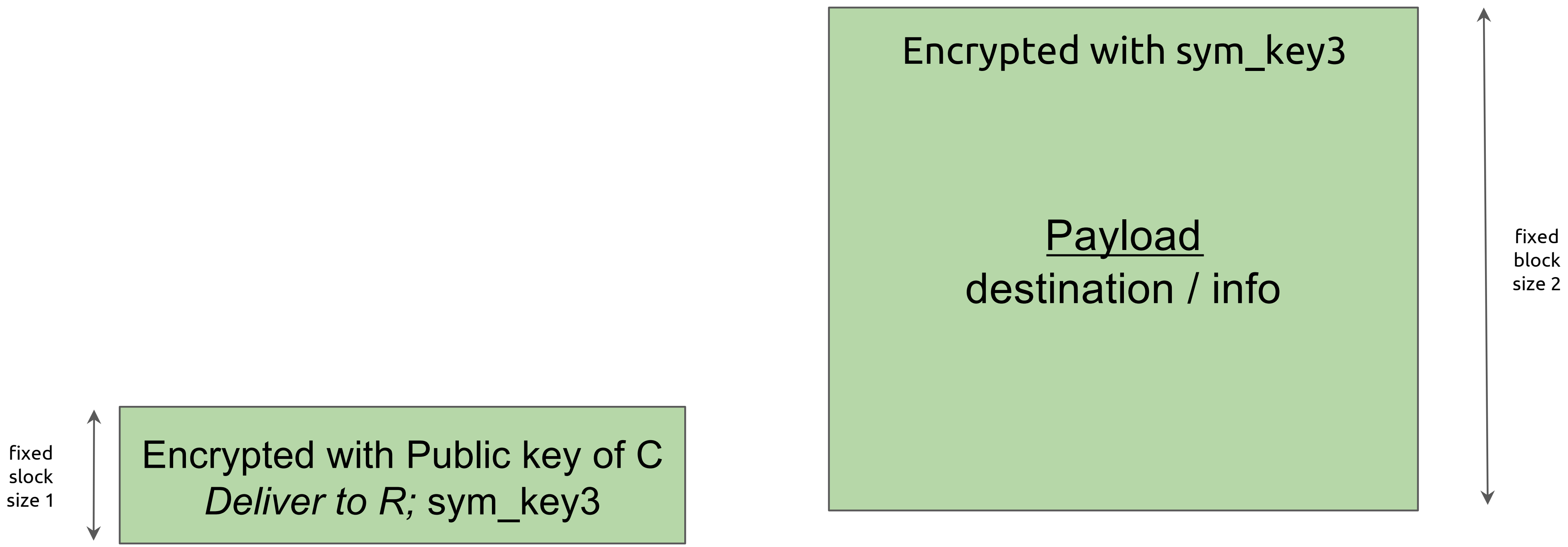
- A mix-net facilitates a sender-anonymous broadcast
- Decryption mix-nets
- Re-encryption mix-nets

Mix-net

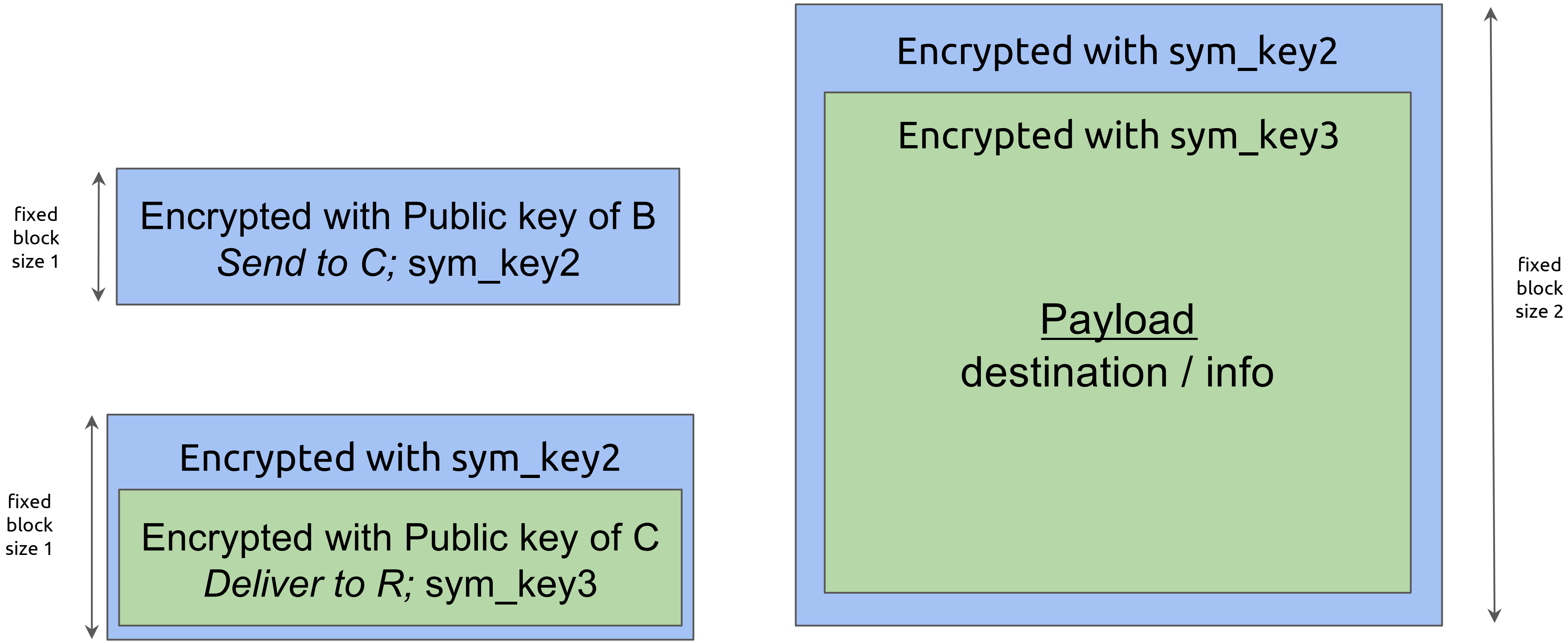


Not possible to relate if S_1 sent m_1 or m_2 (and vice versa for S_2) - as long as there is one hornest mix.

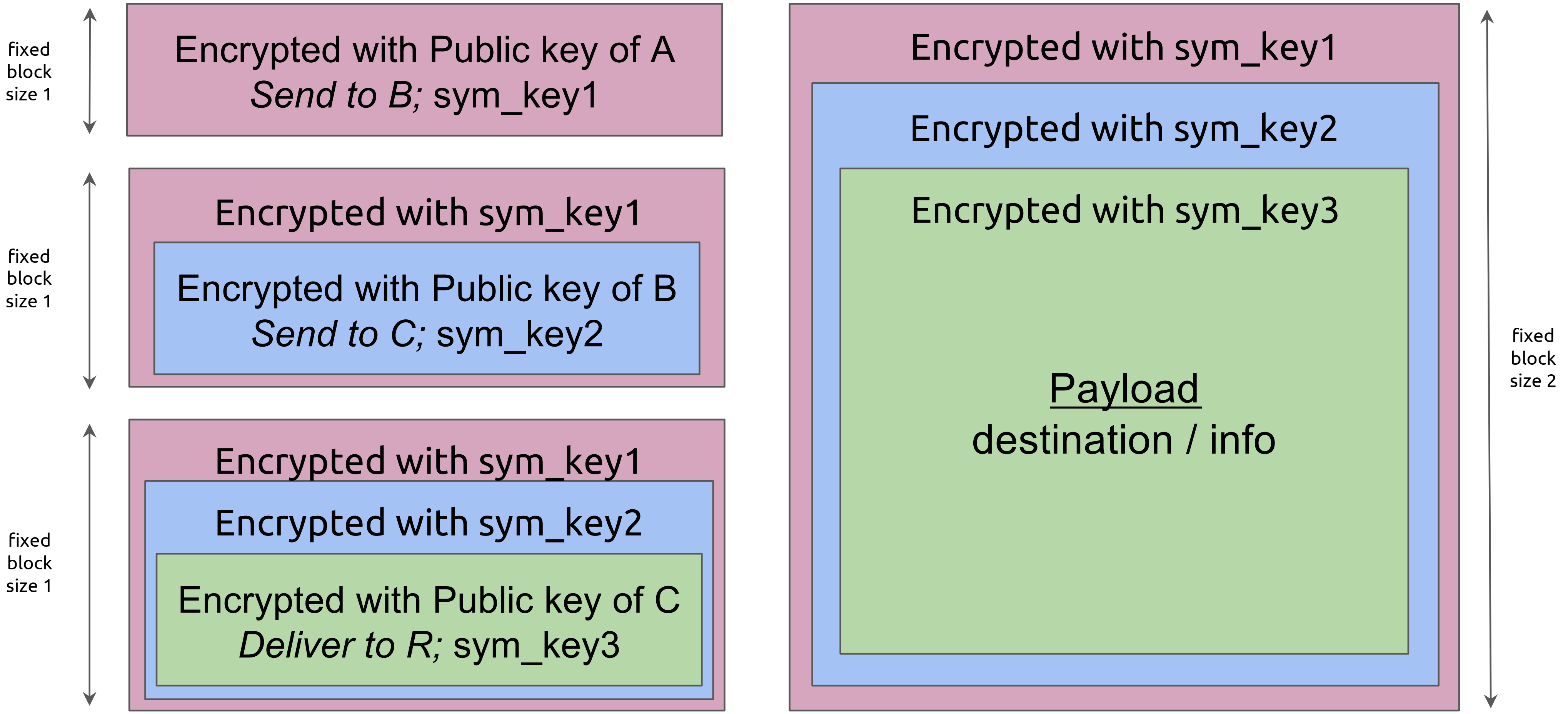
Decryption Mix-net



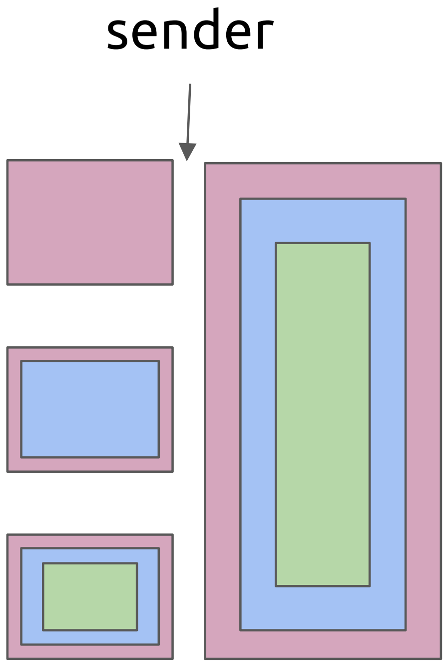
Decryption Mix-net



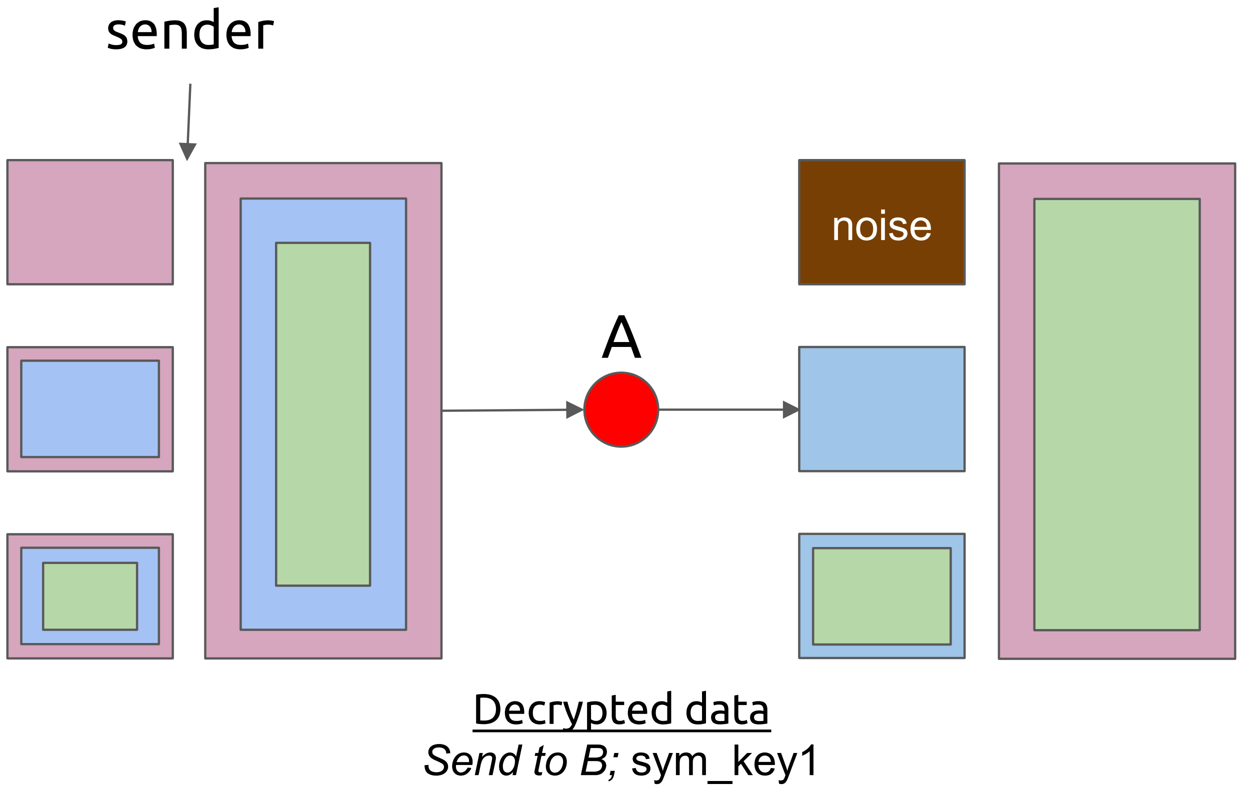
Decryption Mix-net



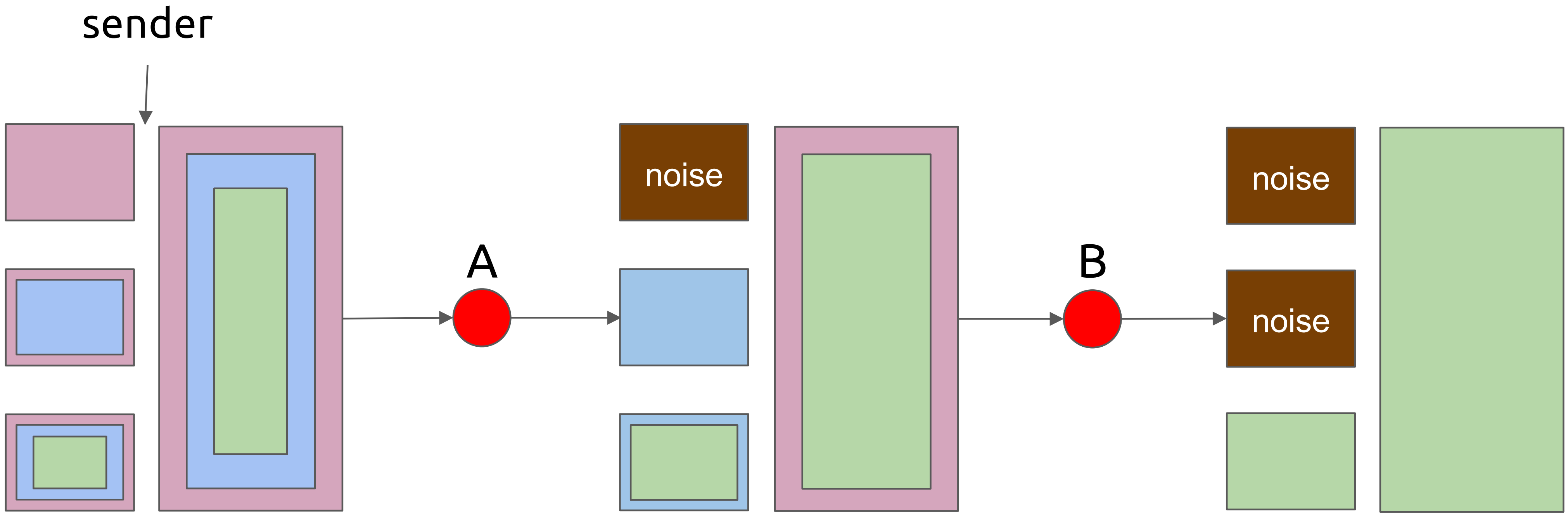
Routing via a Mix-net



Routing via a Mix-net

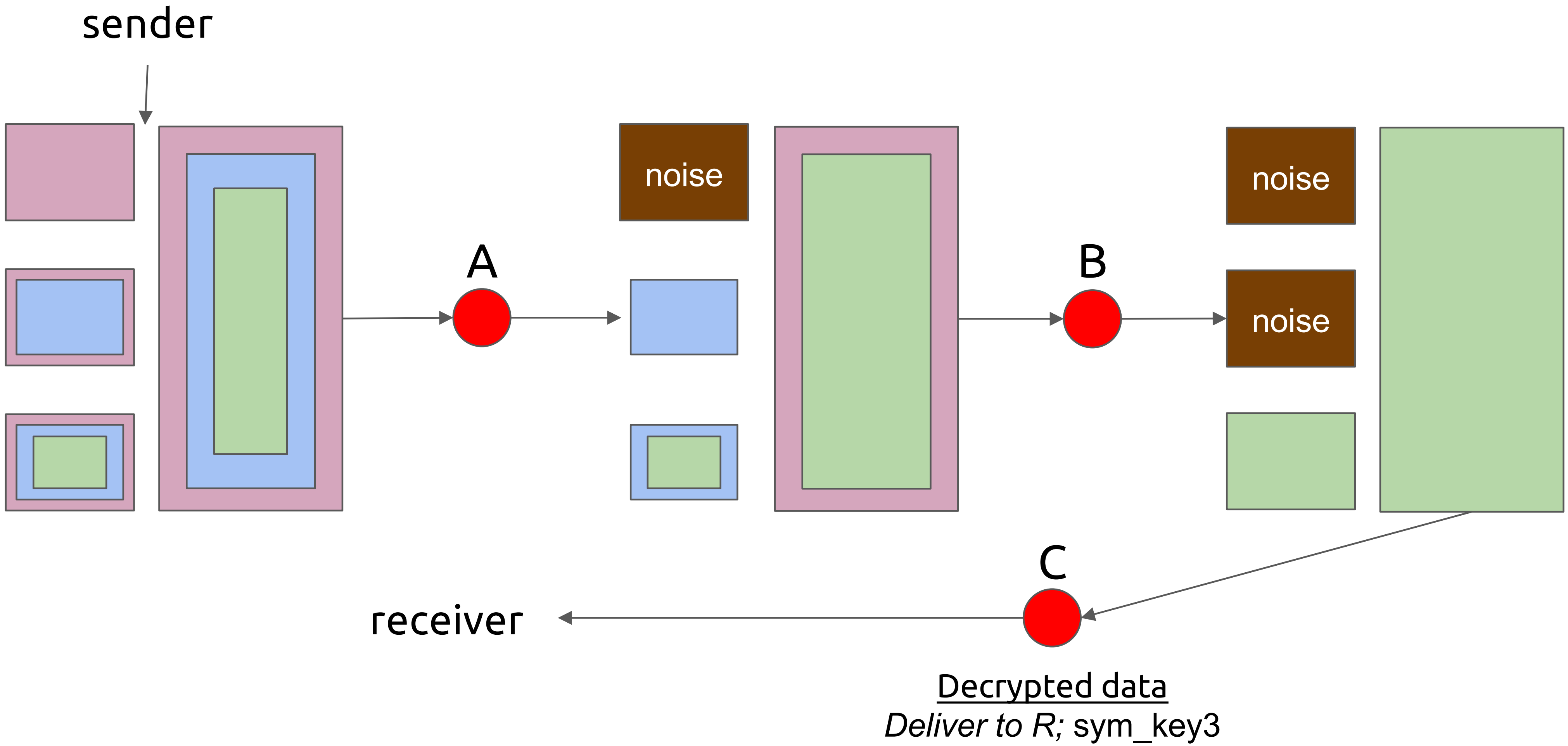


Routing via a Mix-net



<u>Decrypted data</u>
Send to C; sym_key2

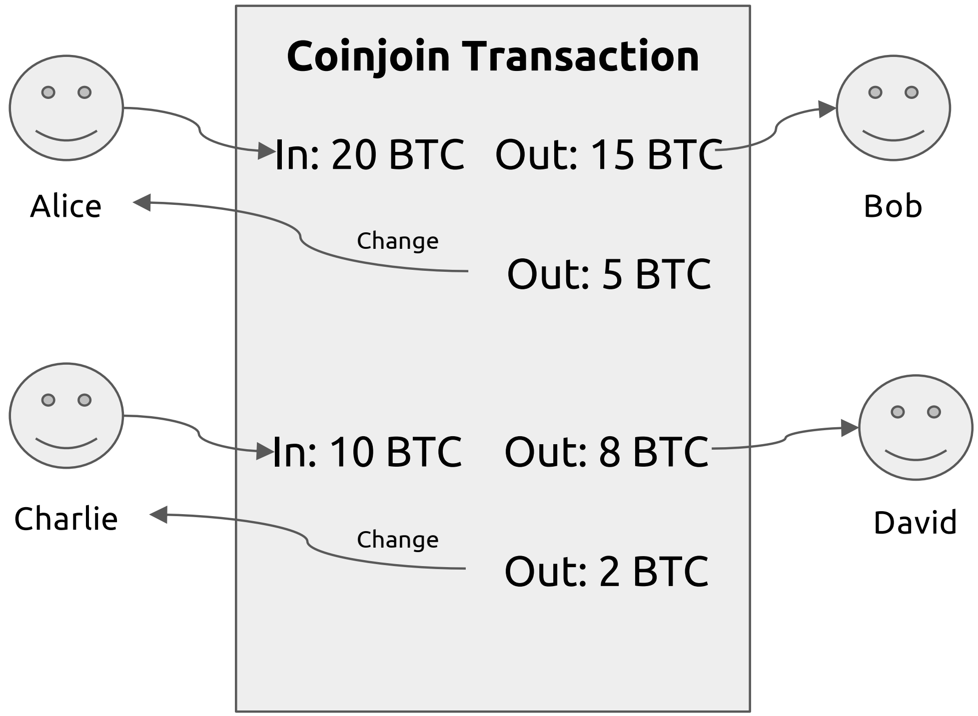
Routing via a Mix-net



Mix-net for Coinjoin transactions

- Parties share with all parties their public-keys (PKI setup)
  - the association between public-keys and accounts a_1, a_2, …, a_n is public
- Parties engage in a decryption mix-net in sequence
  - the **last party** is the **leader**
  - obtains all the relevant information to assemble the multiple input transaction
  - the tx is then sent to all parties
- Note that each step is performed by a designated party P_i
  - any **abort** is **identifiable** and can be attributed to P_i
  - a repeat session may exclude the offending party P_i
- Parties send their signatures to the leader

Hiding Coin Balances



Balances are visible!

Mimblewimble

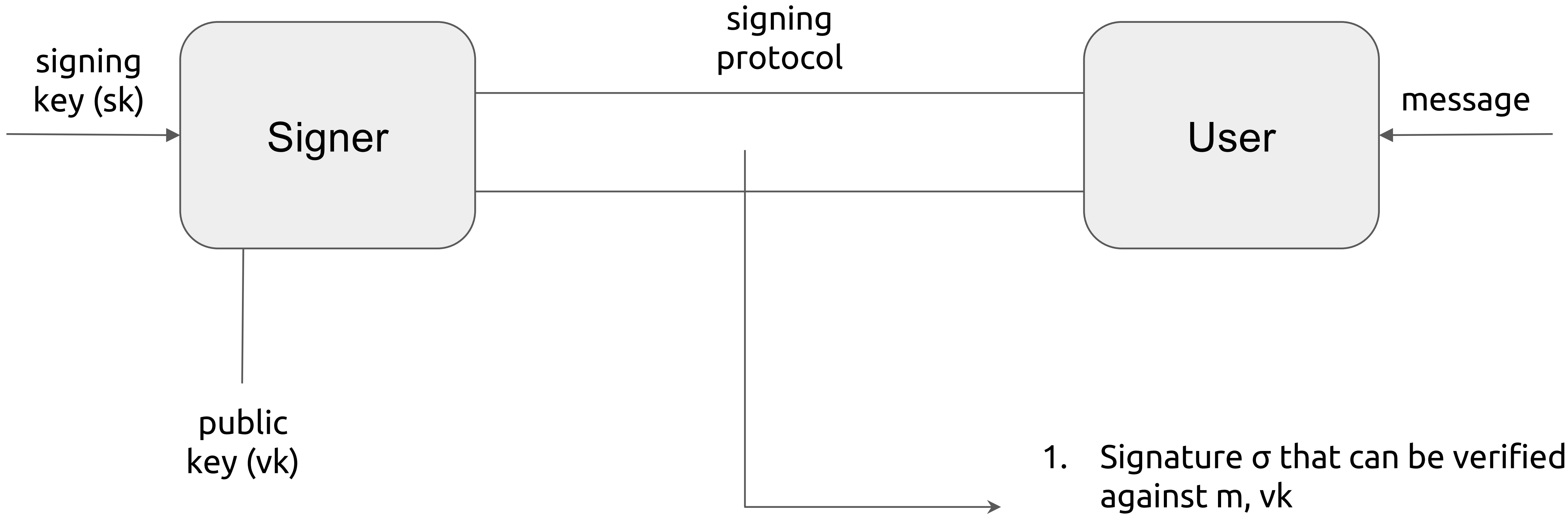
- Use *commitments with homomorphic property*:
  - Com(x) * Com(y) = Com(x+y)
- Instead of revealing the balance transferred, commit to it
  - Pedersen commitment
  - Hiding: commitment does not reveal any information about the value
  - Binding: user cannot open/reveal a value other than the committed
- Ensure value preservation value via the homomorphic property
  - Com(x) * Com(-x) = Com(0)

Georg Fuchsbauer, Michele Orrù, Yannick Seurin. Aggregate Cash Systems: A Cryptographic Investigation of Mimblewimble

Coordination

- CoinJoin and similar techniques require:
  - Coordination
  - Message passing between multiple parties
- How do parties find each other?
- How to prevent DoS attacks?
- Is it possible to improve with more advanced cryptographic techniques?

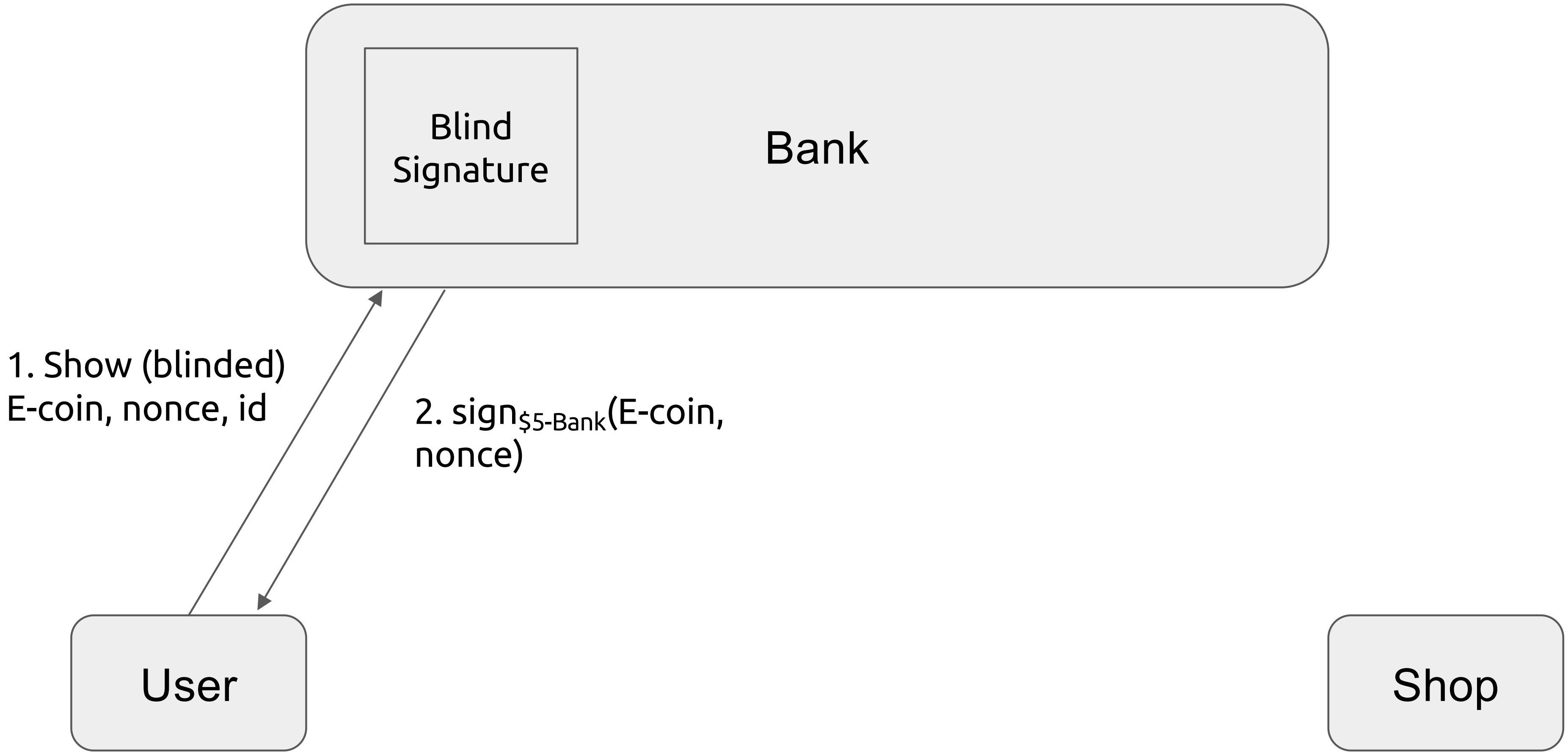
Blind-Signatures



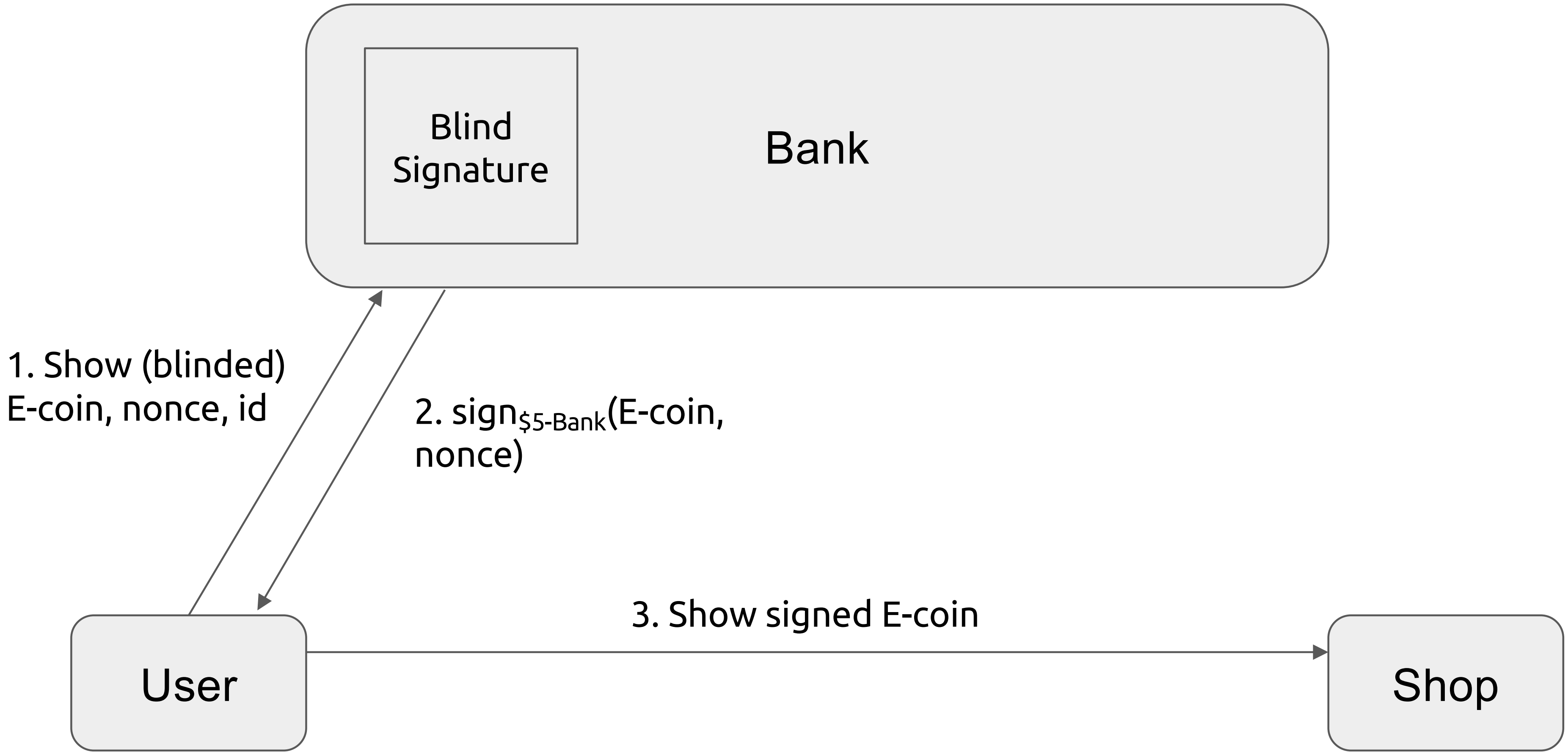
1. Signature σ that can be verified against m, vk
2. Signer doesn’t see the message
3. Signer cannot link two published (m, σ) with which users requested them

Chaum’s E-cash

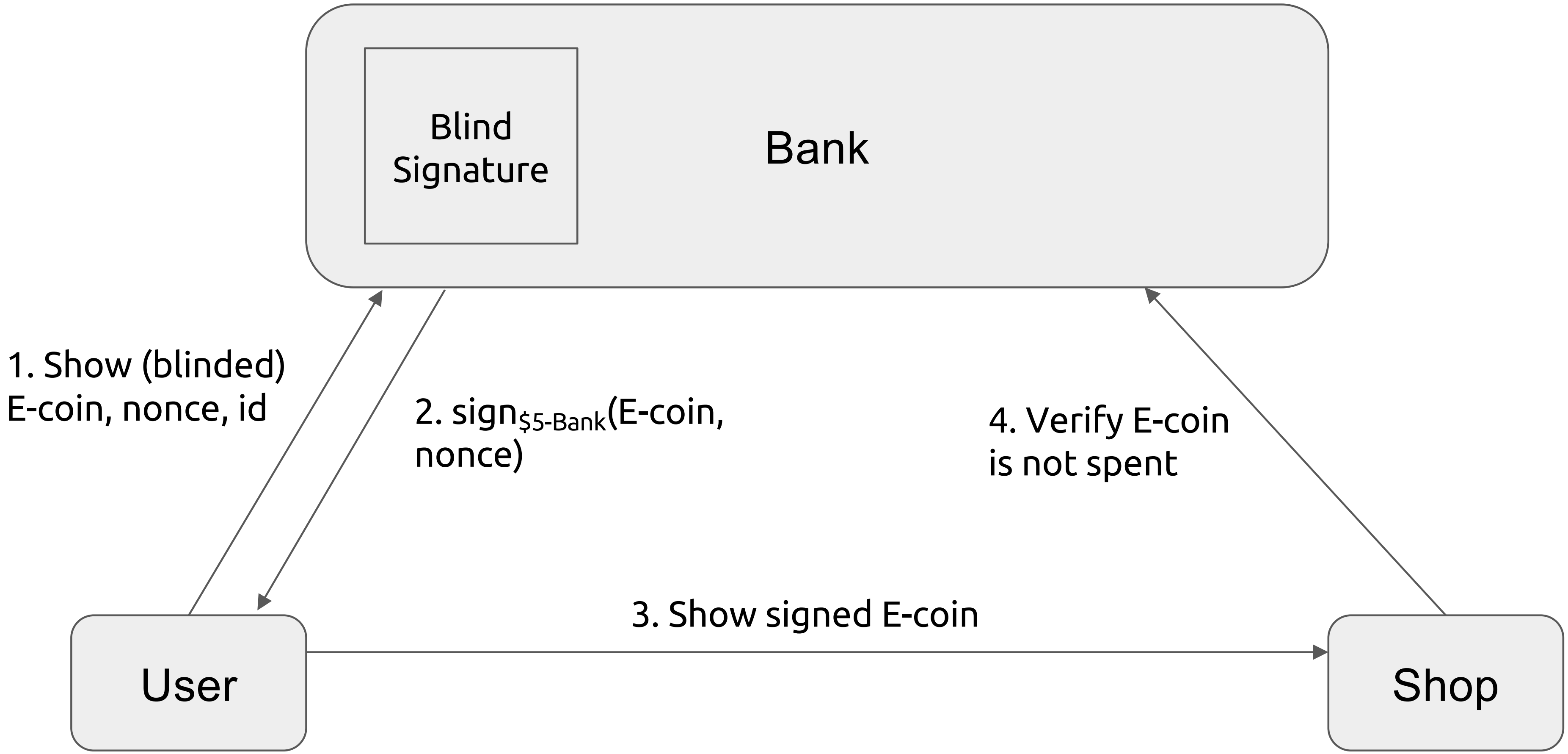
Chaum's E-cash



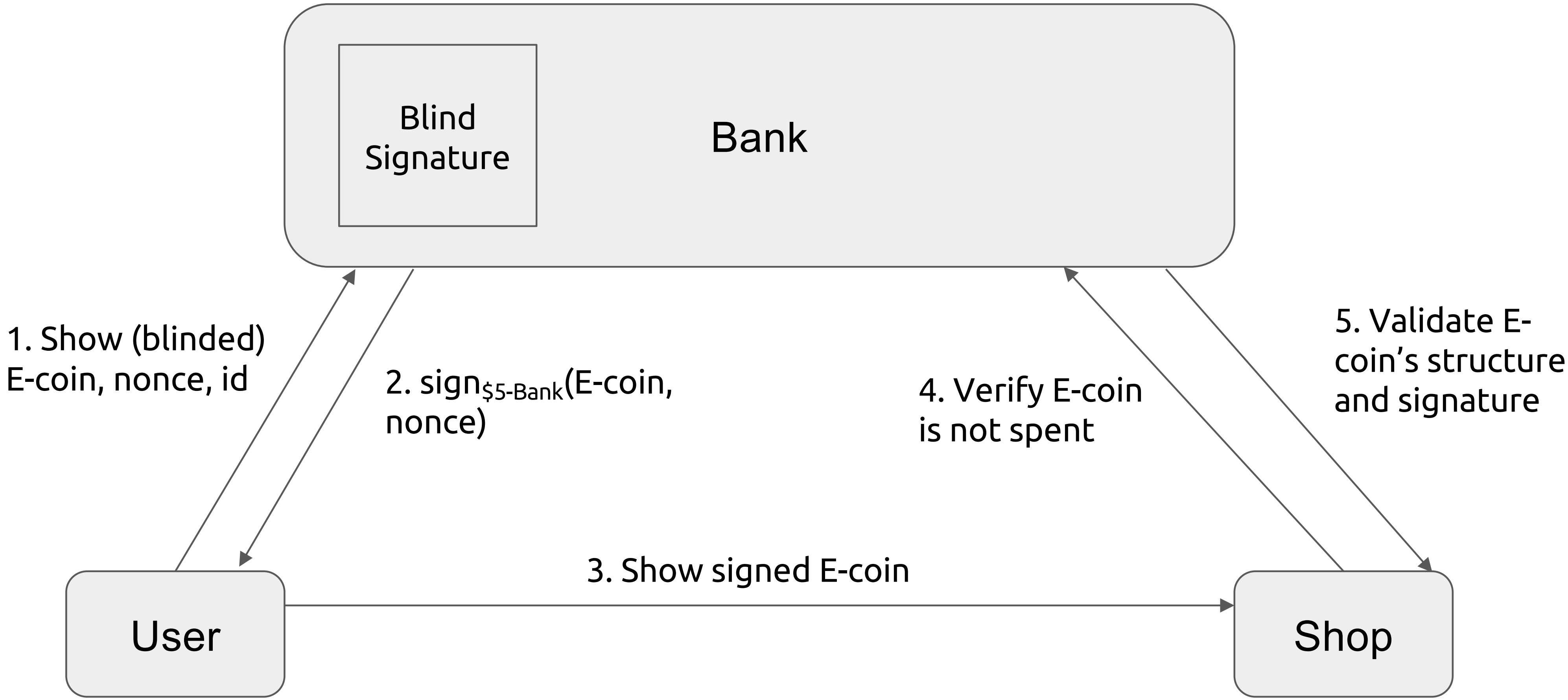
Chaum’s E-cash



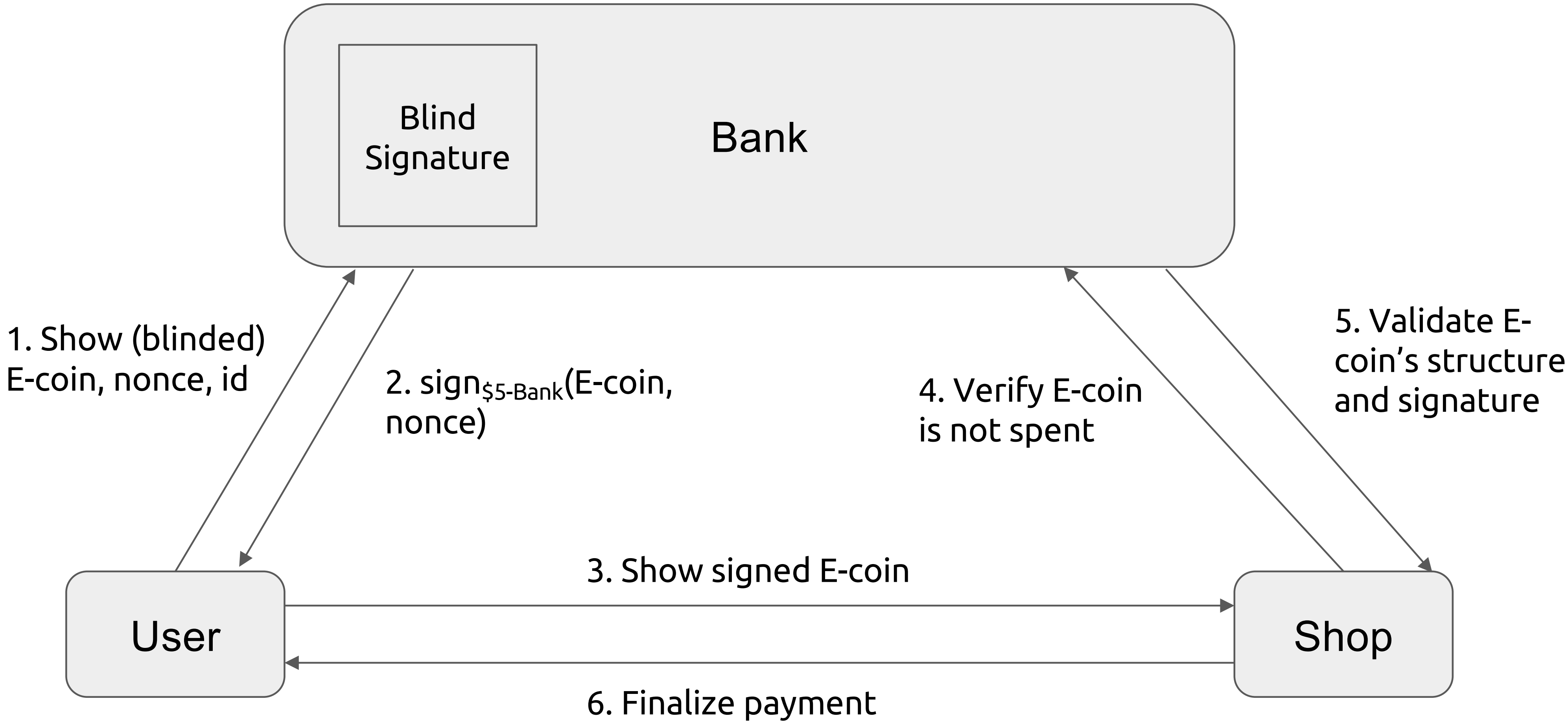
Chaum’s E-cash



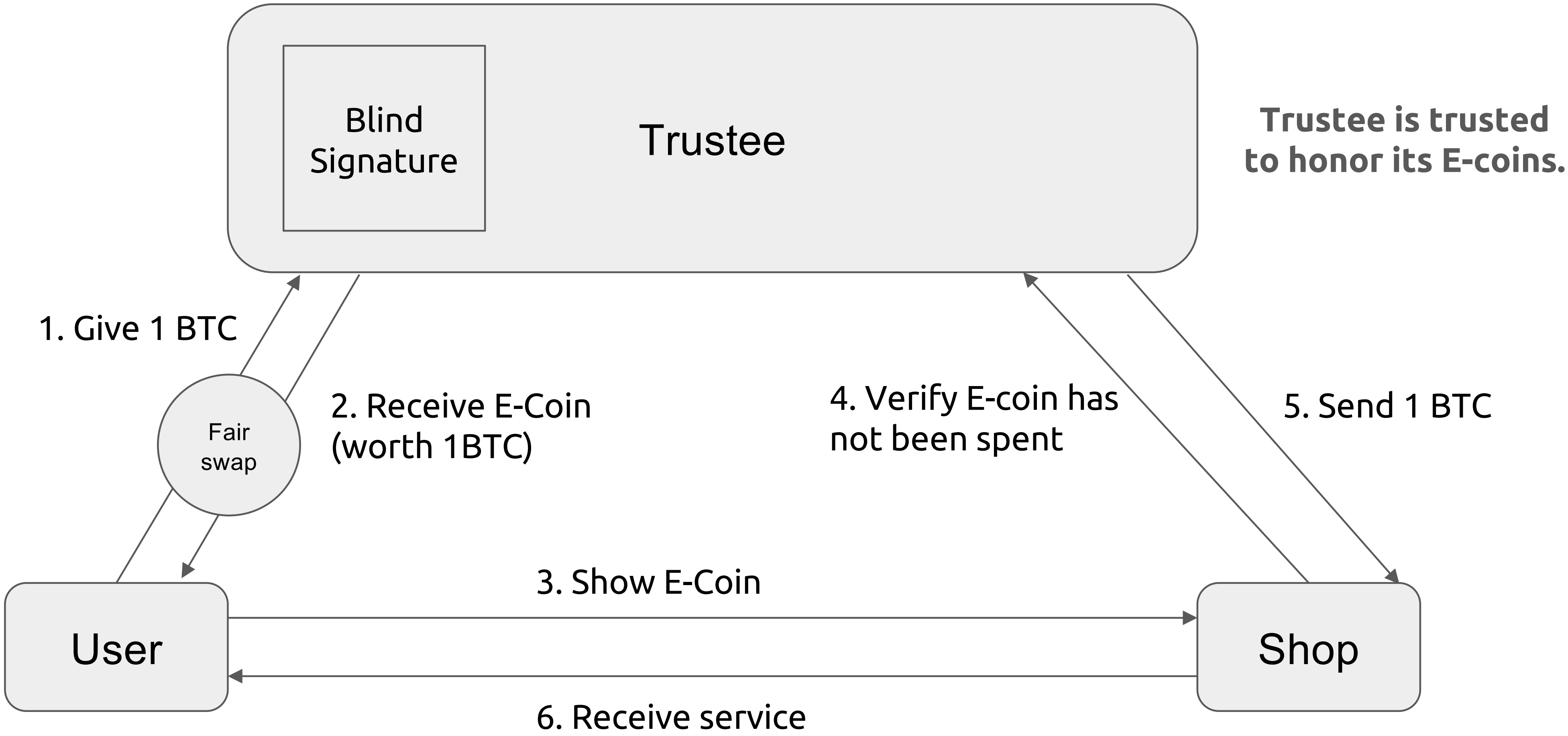
Chaum's E-cash



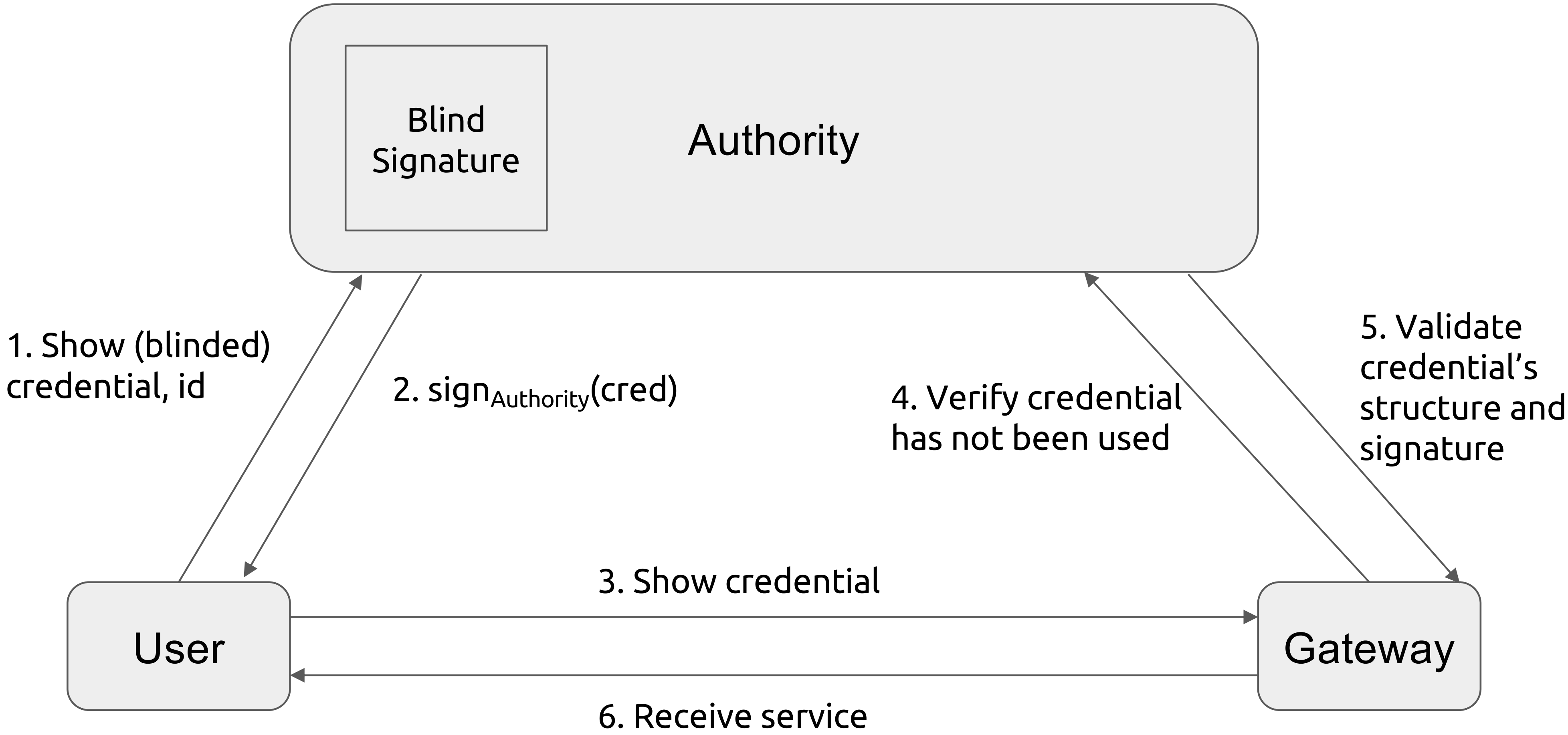
Chaum's E-cash



Anonymizing Bitcoin Payments via E-cash



Anonymous Credentials



Fair Swaps

- Alice and Bob would like to exchange secrets s.t.:
  - either none of them gets their output
  - or both do
- Classical problem
- Impossible to solve under standard network assumptions!
  - (intuition: one party always has the ``upper hand.'')
- Going around the impossibility:
  - optimistic fair exchange
    - Resort to a trusted third party, when things go bad.
  - resource-based fair exchange
    - Parties exchange puzzles in multiple rounds, so if one aborts and they can recover the other's secret, the latter can also do that with a bit more effort.
  - fair swaps with penalties
    - In case of an abort, the aborting party has to pay a penalty.

Fair Swaps - Construction

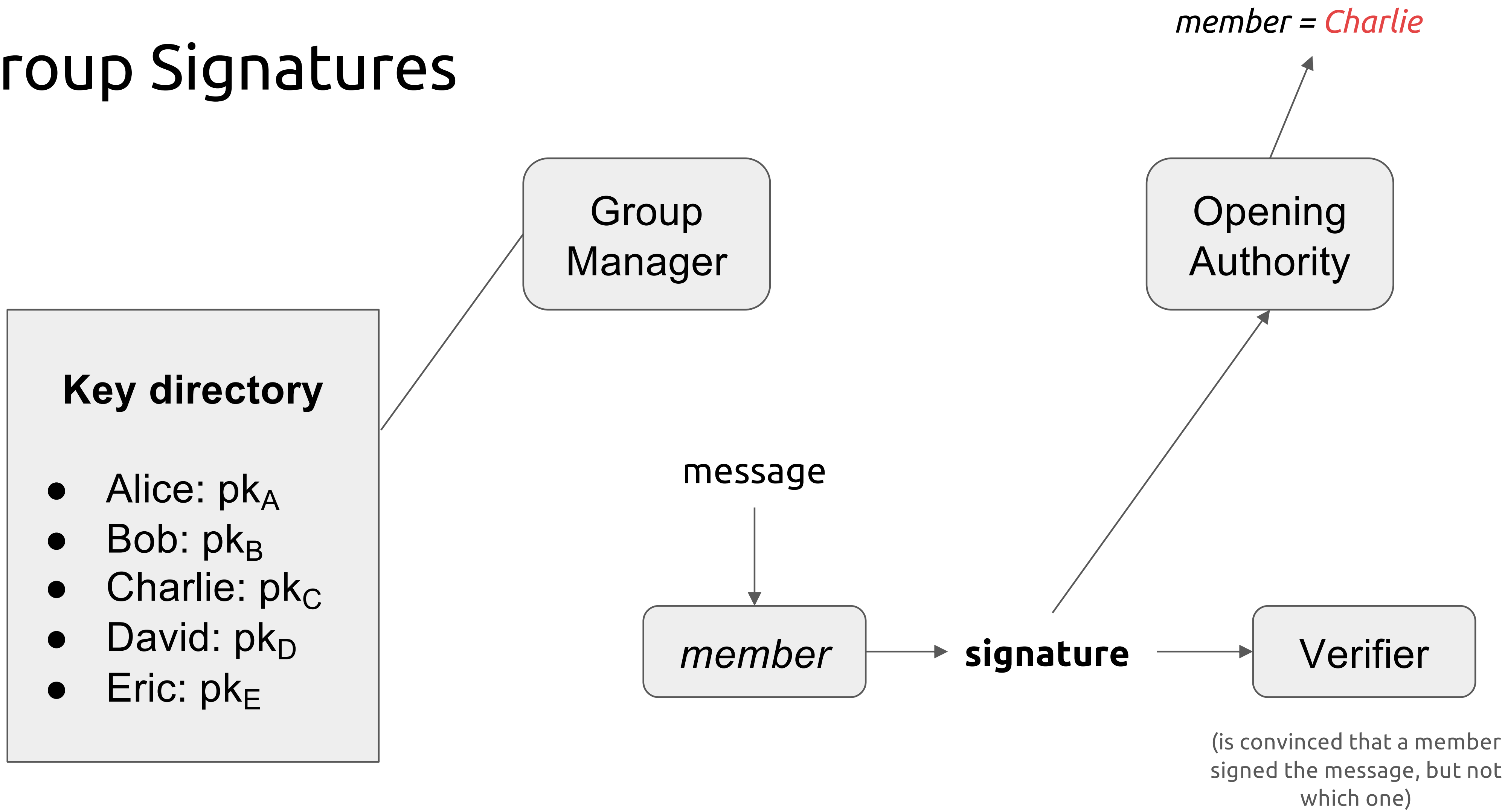
- Using a blockchain that supports **smart contracts**
- A contract that both parties fund to accept their secrets
- Key requirements:
  - parties lock up some funds in **deposits**
  - secret submission should be **verifiable** by the contract's code
- Fair swap with penalty:
  - Either both parties get their output
  - Or the offending party is **penalized financially**
- Easier problem: Fair swap of tokens
  - Parties lock the required number of tokens in a smart contract that facilitates the swap in case all values are in. If not, it allows parties to withdraw after a timeout.

Anonymity and Digital Signatures

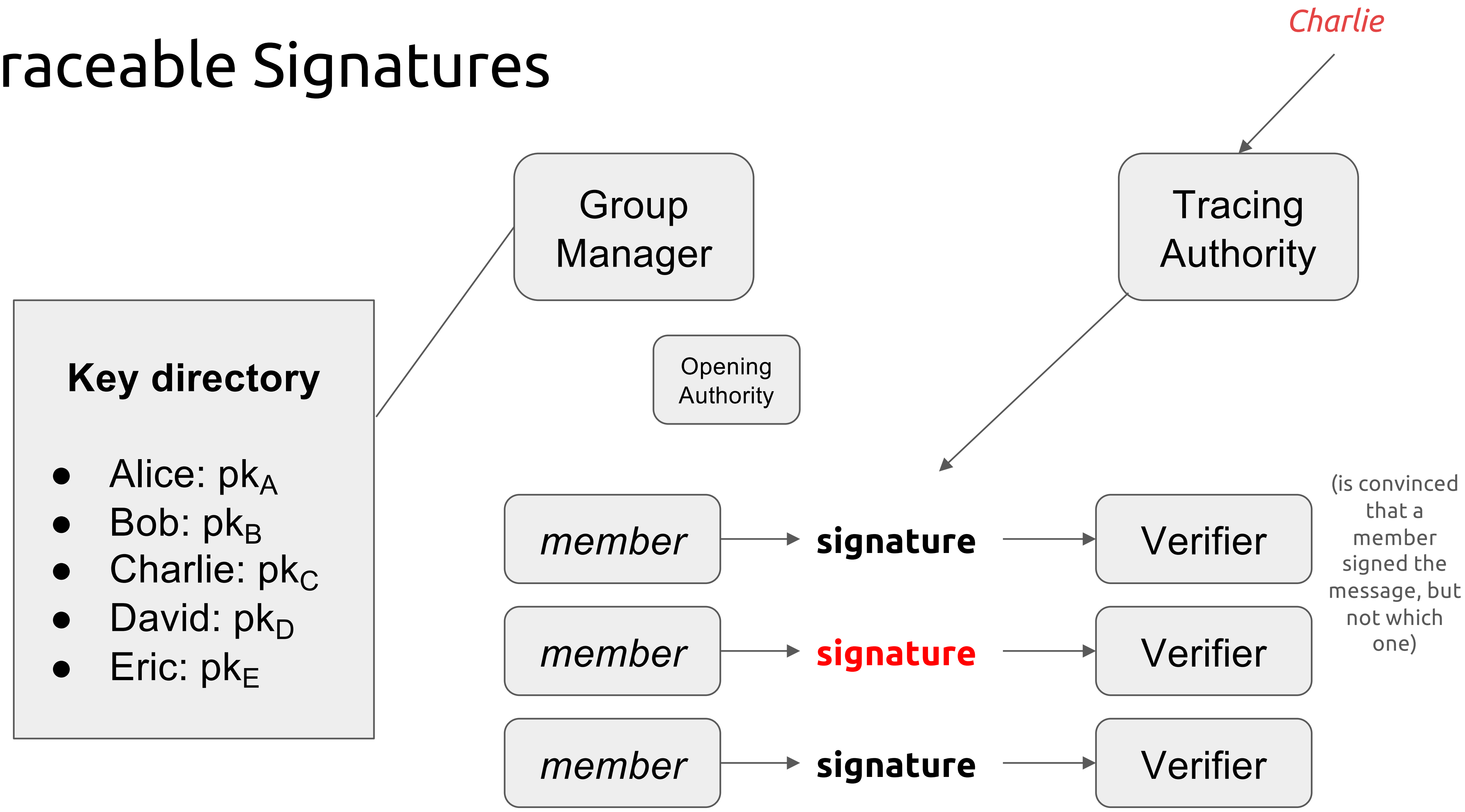
Anonymity and Digital Signatures

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

Group Signatures



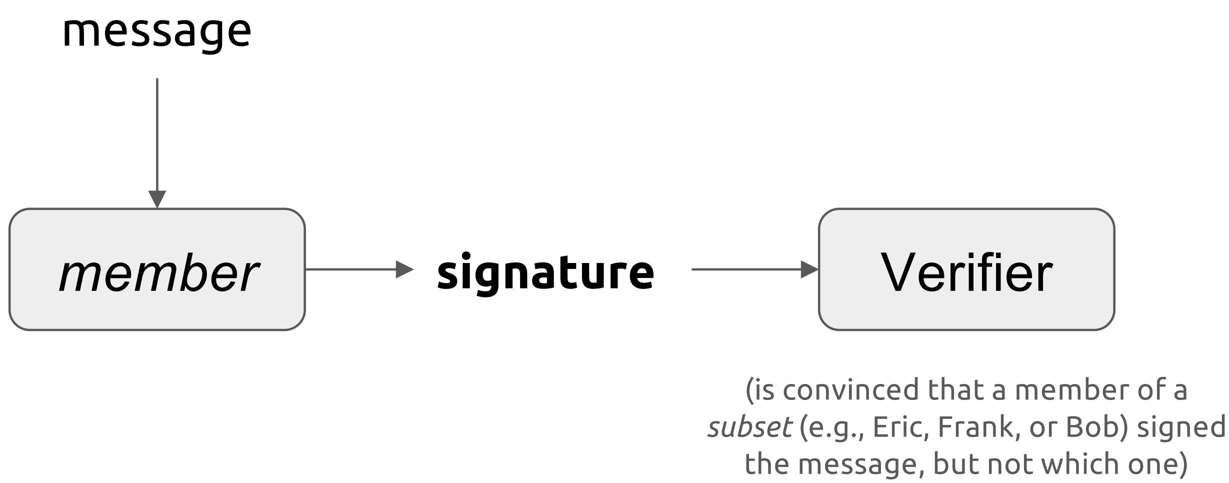
Traceable Signatures



Ring Signatures

Key directory

- Alice: pk_A
- Bob: pk_B
- Charlie: pk_C
- David: pk_D
- Eric: pk_E



Monero/Cryptonote

- **Linkable ring signatures**
- **“Stealth” addresses**
- For each payment, an anonymity set is selected with accounts of the same monetary value
- A ring signature is issued on behalf of that set:
  - suitably restricted s.t. an account can only be used once
  - if an output is used twice, it is *linkable* and certain inferences can be made
- Stealth addresses enable:
  - the sender to create unlinkable addresses for the receiver
  - the receiver to detect said addresses

Is Monero Anonymous?

- There is potentially more uncertainty in the Monero blockchain compared to a Bitcoin-like blockchain (even with Coinjoin transactions)
- However, it is not obvious how to **quantify the level of anonymization**
- **De-anonymization** is **feasible** in reasonable real-world threat models
  - e.g., the attacker "sprays" the ledger with transactions s.t. it commands a good number of selected accounts

The revelance of the anonymity set

Dec 18, 2013, 01:46pm EST

Harvard Student Receives F For Tor Failure While Sending 'Anonymous' Bomb Threat

Runa A. Sandvik Former Contributor

Tech

I cover all things privacy, security and technology.

Follow

According to the five-page complaint, the student "took steps to disguise his identity" by using Tor, a software which allows users to browse the web anonymously, and Guerrilla Mail, a service which allows users to create free, temporary email addresses.

(Photo credit: joeythibault)

What Kim didn't realize is that Tor, which masks online activity, doesn't hide the fact that you are using the software. In analyzing the headers of the emails sent through the Guerrilla Mail account, authorities were able to determine that the anonymous sender was connected to the anonymity network.

Using that conclusion, they then attempted to discern which students had been using Tor on the Harvard wireless network around the time of the threats. Before firing up Tor, Kim had to log on to the school's

Given how quickly he was found, Kim was likely one of the few—if not the only—individuals on Tor around on Monday morning. According to authorities, he "anonymously" emailed threats including ""bombs

Increasing and Safeguarding the anonymity set

- A **larger anonymity set** is most **preferable**
- In the techniques seen so far, transaction preparation work **increases linearly** with the anonymity set
- **Ultimate Goal:** use the set of all possible Unspent Transaction Outputs (UTxOs)

Increasing and Safeguarding the anonymity set

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

- *sn*: the serial number of a valid $1 coin
- ρ: random
- The commitment value is **associated** with a ledger deposit
  - A coin for $1 is “minted”
- Spending a coin requires **announcing** *sn* and **proving** it had been committed in the ledger:
  - existential quantifier over all ledger commitments
  - $\exists i : \psi_i = \mathrm{Commit}(\rho, sn)$

Increasing and Safeguarding the anonymity set

- Organize all commitments and serial numbers in a Merkle tree
- Prove that there is a leaf in the Merkle tree that contains the commitment
  - ψ_l = Commit(ρ, sn)
- Statement representation and witness size logarithmic in the number of coins

Increasing and Safeguarding the anonymity set

- Organize all commitments and serial numbers in a Merkle tree
- Prove that there is a leaf in the Merkle tree that contains the commitment
  - ψ_l = Commit(ρ, sn)
- Statement representation and witness size logarithmic in the number of coins
- Challenges
  - How to prove efficiently a statement referring to the leaf of a Merkle tree?
    - 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

Zero-knowledge

ZK-Snarks

- Zero-knowledge succinct arguments of knowledge
- Similar to “zero-knowledge proofs”
- Can prove possession of a witness for any public statement / predicate
- **Computational soundness**:
  - depends on the security of a “common reference string” (a structured cryptographic information that is assumed to be honestly sampled)
- **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

$$\langle vk, v, \rho \rangle$$
$$k = \text{Commit}(s, vk||\rho)$$
$$sn = \text{PRF}_{\text{sk}} (\rho)$$
$$vk = \text{PRF}_{\text{sk}} (0)$$
$$\psi = \text{Commit}(s', v||k)$$
$$\text{coin: } (vk, v, \rho, s, s', \psi)$$

- *(vk, sk)*: account's public/private key
- *v*: value, *ρ*: random (seed for serial number), *s, s'*: random (nonces for commitments)
- PRF: pseudorandom function
- Commit: commitment function

The double commitment enables verifying that the value *v* is properly encoded in the coin (within *ψ*) without revealing *vk,* or *sn* (which is inside k)

Zerocash "Pour" Operation

- Given a coin: *(vk, v, ρ, s, s', ψ)*
- Produce two new coins with values $v_1 + v_2 = v$
    - Use keys vk_1, vk_2
    - Serial number *sn* of spent coin is revealed and marked as spent
- Set:
    - k_i = Commit(s_i, $vk_i||\rho_i$), i=1, 2
    - ψ_i = Commit(s_i', $v_i||k_i$), i=1, 2
- Show ψ_1, ψ_2 and *sn* and prove
    - ψ_1, ψ_2 are well formed
    - Prove Merkle tree contains some coin *(vk, v, ρ, s, s', ψ)*
    - Check that serial number satisfies $sn = PRF_{sk}(\rho)$ and $vk = PRF_{sk}(0)$
    - Value is split properly $v_1 + v_2 = v$, against the committed values in ψ_1, ψ_2
- Encrypt s_i , s_i' , ρ_i with the key of the recipient of the coin.
- System checks *sn* is not spent and proof is correct.

Common Reference Strings

- SNARKs require a “common reference string”
- A **trusted computation** is needed to produce it
  - Use **secure multiparty computation (MPC)**
  - Use **updateable reference strings (URS)** instead and outsource the update operation to miners/blockchain participants
  - Use **alternatives to SNARKs** that do not require it
    - Disadvantage: worse performance
    - e.g., Bulletproofs

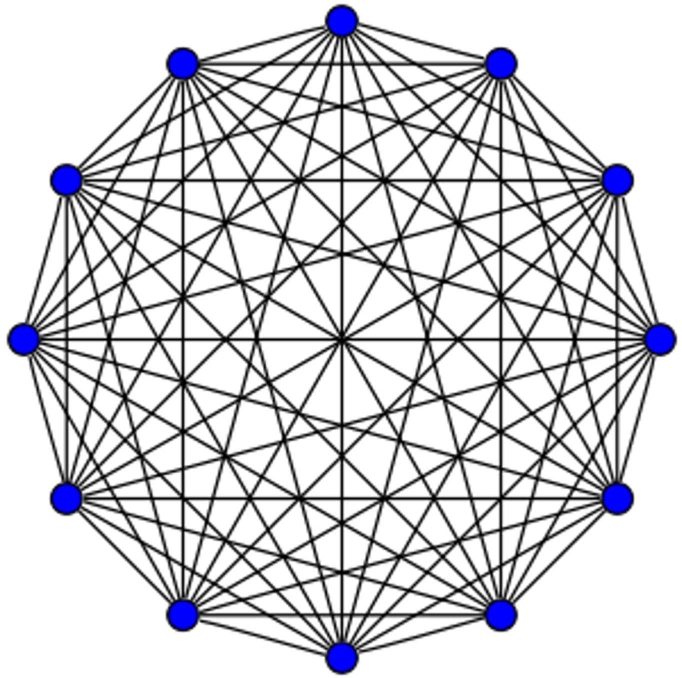
Network security

Overlay Networks

- A reliable network is critical for blockchains and distributed ledger protocols to operate
- Typically they utilize an **overlay network**
  - a network built on top of another network
  - virtual links connect the participating nodes

Overlay Networks

in a network, we would like nodes to be fully connected



relevant operations :

1. point-to-point communication
2. broadcast

Network Requirements

- Synchronicity
- Reliable message transmission
- Reliable Broadcast

Bitcoin’s P2P Network

- A **Peer-to-Peer** (P2P) network over TCP/IP
- Peers are identified by their IP address
- Peers can **diffuse** messages to be propagated to the whole network
- Peers initiate a **small number** of **outgoing** connections
- Peers receive a limited number of **incoming** connections

Public vs. Private networks

- A system with a public IP “lives” in the Internet
- A system with a private IP “lives” in a private network and communicates with the Internet via a router that performs Network Address Translation (NAT)

P2P Networks

- (In the case of Bitcoin) The requesting node contacts a **DNS Seeder**:
  - A node with a public IP address that can serve a list of IP addresses for Bitcoin nodes
  - Obtains those addresses via **crawling**
- If the connection fails, the node has a hardcoded set of IP addresses
- Peers exchange node IP addresses via **ADDR messages** that contain a selection of a peer’s address book

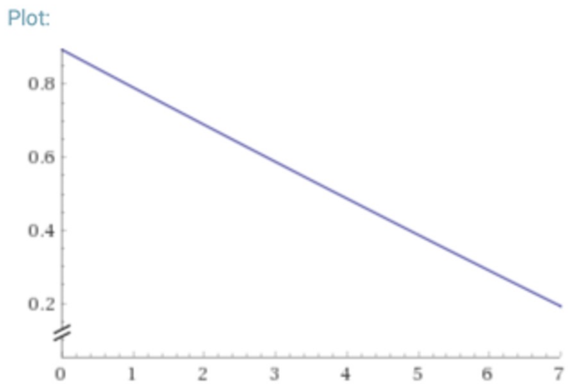
Table maintenance

- Nodes maintain tables of peers that they have learned:
  - Nodes that have proven to be operational
  - Nodes for which the node has been informed about their existence, but they have not been contacted yet
- Tables are updated on a regular basis
- Timestamp information is stored from the last connection attempt

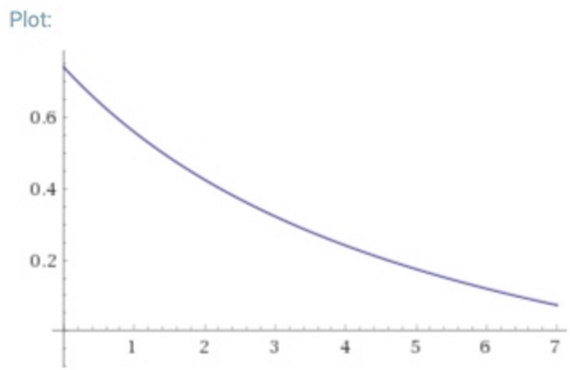
Connect to new or tried peers?

- Tables “new” and “tried”
- A node with $\omega \in \{0, ..., 7\}$ outgoing connections will select the $\omega+1$ connection from **tried** with probability: $\frac{\sqrt{\rho}(9-\omega)}{(\omega+1)+\sqrt{\rho}(9-\omega)}$
  - ρ: ratio between #(addresses in tried) and #(addresses in new)
- Choose from the selected table an address to connect, biasing towards addresses with fresher timestamps

ρ=0.9



ρ=0.1



Attacking the P2P layer - Key Observations

- A node will add an address to the ‘tried' table if it receives an incoming connection from another node
- A node will accept unsolicited ADDR messages; these will be added to the ‘new’ table
- Nodes rarely solicit information from DNS seeders and other nodes

Eclipse Attack

- Victim is a node with a public IP
- Attacker makes outgoing connection to the node using adversarial nodes
  - ‘tried’ table gets full with fresh adversarial IP’s
- Attacker uses ADDR messages to insert trash IP’s into the ‘new’ table of the victim
- Attacker waits for the victim node to restart (nodes maintain existing outgoing connections)
  - Restarts can happen because of a software update or even deliberately by the attacker via a DOS attack

Eclipse Attack

- The attacker can repetitively connect to victim node to ensure timestamps of adversarial nodes are fresh
- If a ‘new’ address is selected:
  - injection of trash IPs ensures that, with some probability, the new node will not be responsive
  - another coin flip will be attempted for the connection, which can result to an adversarial IP

Eclipse Attack

- Attacker saturates the incoming connections of the victim
  - The protocol allows for the same IP to occupy all 117 incoming TCP/IP connections
- It becomes impossible for other nodes to connect to the victim
- As maximum number of connections is reached, the victim will deny any other incoming connections

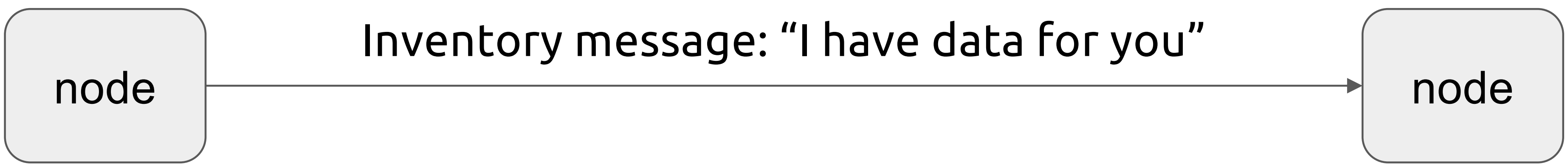
Eclipse Attack

- Once the eclipse takes place, all (incoming/outgoing) communication of the victim is routed via the attacker nodes
  - victim’s transactions may be censored
  - victim’s blocks can be dropped
  - victim’s blockchain could be populated almost entirely by adversarial blocks!
- The rest of the network will eventually completely forget about the victim node
  - a function *isTerrible* is executed periodically on the tables to remove any node that has an over-30-days old timestamp and too many failed connection attempts

Attack Countermeasures

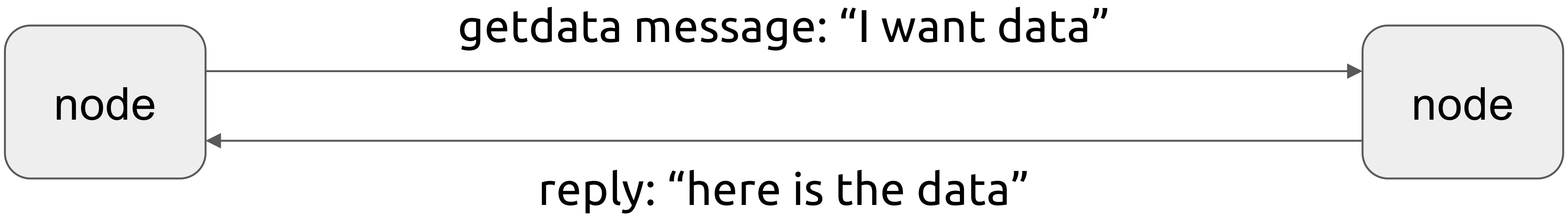
- A number mitigation techniques can be used:
    - ban unsolicited ADDR messages
    - Diversify / validate incoming connections
    - test before evicting addresses from the tried table

Information propagation in Bitcoin



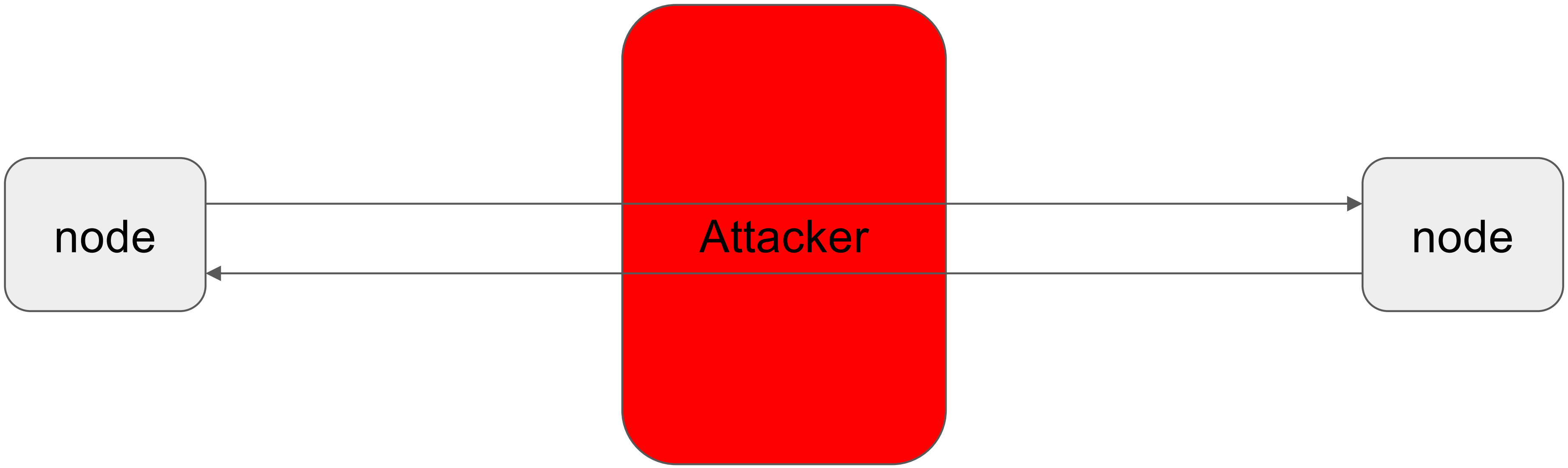
Field Size	Description	Data type	Comments
4	type	uint32_t	Identifies the object type linked to this inventory
32	hash	char[32]	Hash of the object

Information propagation in Bitcoin



20-minute window before connection is dropped

Man-in-the-Middle attacks



- If attacker manipulates message contents on either direction, it can delay information propagation by 20 minutes.
- Such delays can be extremely detrimental for security

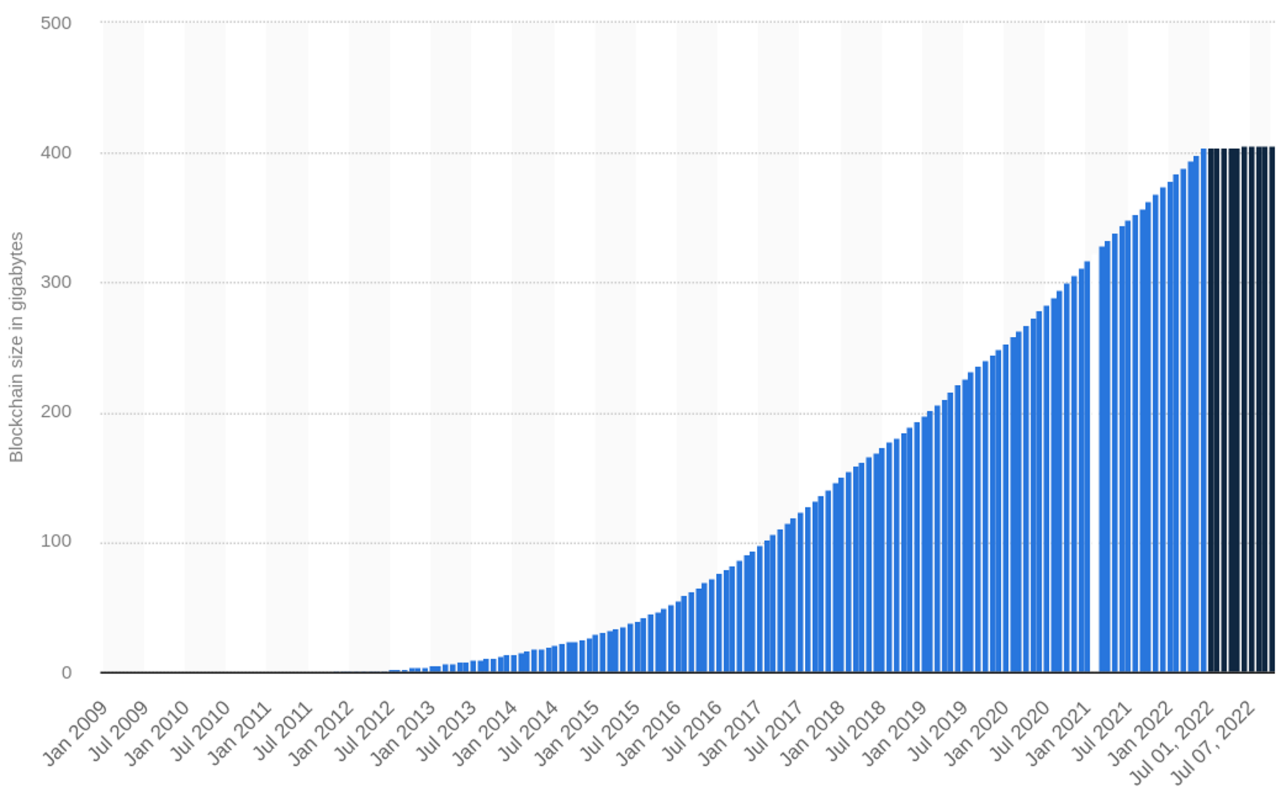
Wallets

Full nodes

- Some wallets maintain the whole blockchain
- Full nodes:
  - Keep the whole blockchain history (~187 GB)
  - Keep the whole UTxO set
  - Verify each tx
  - Verify each block
  - Relay every tx and block

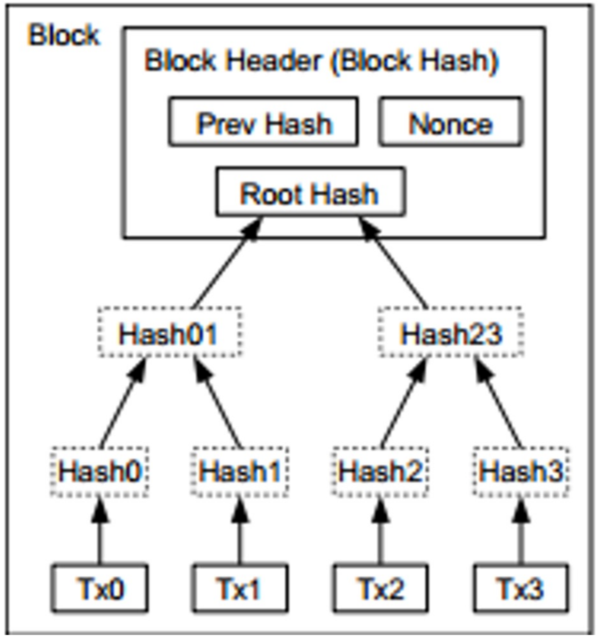
Size of the Bitcoin blockchain from January 2009 to July 11, 2022

(in gigabytes)



Recall : Merkle trees of transactions

- Transactions not yet confirmed, but received by a full node are collected into a data structure called the **mempool**
- To build a block, the mempool transactions **are collected** into a Merkle Tree in an (arbitrary, but valid) order defined by the miner
- The application data in the **block header**, for which the Proof-of-Work equation is solved, only contain the **root** of this Merkle Tree: **x**



Advantages of using a Merkle tree

- Proof-of-Work difficulty **does not depend** on the number of confirmed transactions
  - each miner is incentivized to include all transactions they can, which have a non-zero fee
- The PoW difficulty **only depends on the target T**
  - this allows better control of the mining rate
- It enables **SPV wallets!**

SPV

- Simple Payment Verification
- A different type of wallet
- Useful for mobile, laptops etc.
- Doesn’t need to download the whole blockchain
  - Does not download all transactions
  - Much faster than standard (full) node
- Keeps **only the block headers from genesis till today**
- Connects to multiple **untrusted** servers
- Server is a full node which **proves** to the SPV wallet each claim

SPV

- Wallet sends to the SPV server the bitcoin addresses they have
  - Not the private keys!
  - The SPV server knows which transactions to send to the SPV client
  - The addresses are shared via a Bloom filter
- Wallet **verifies** each block's **PoW** and authenticated ancestry
  - Keeps a longest chain as usual
  - Does not keep transactions
- Wallet verifies **each transaction** it receives
  - Signatures
  - Law of conservation
- Wallet verifies that the transaction belongs to the Merkle Tree root of a block

SPV Security

- SPV wallets
  - **don’t keep** a UTXO
  - **don’t verify or receive** transactions they are not interested in
  - **don’t verify** coinbase validity
- Have the *same level of security* as a regular full node
  - assuming honest majority
- What can a malicious SPV server achieve?
  - Temporary fork to invalid block (invalid coinbase, transactions, non-existing UTXO, double spending...)

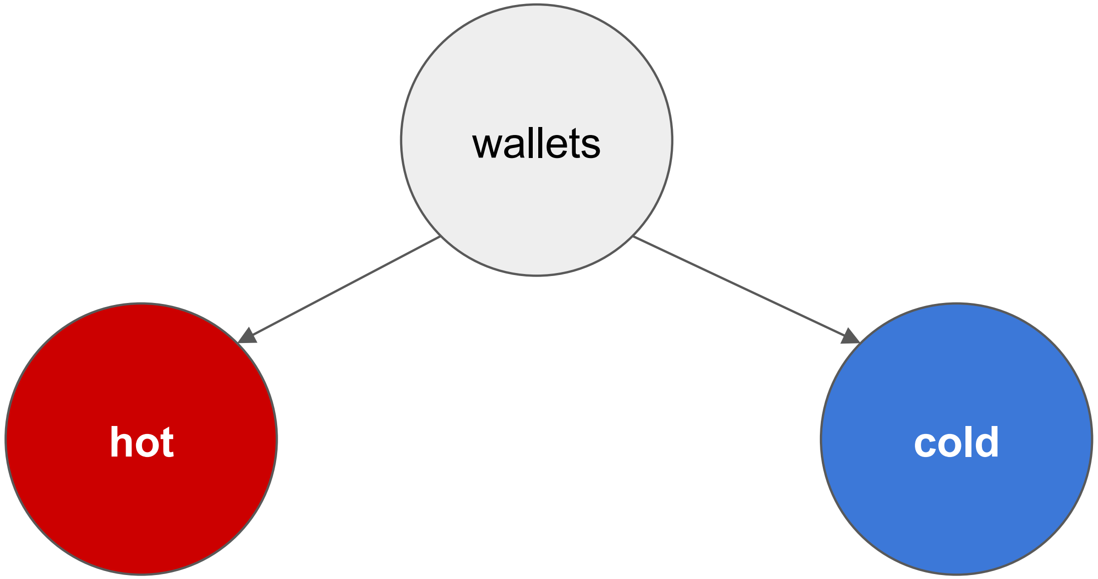
Wallet seeds and HD wallets

- Hierarchical Deterministic (HD) wallet
- An infinite sequence of wallet private keys can be generated from a single “master private key” (BIP-32)
- A private key can be encoded as a human-readable **seed**
- Seed is sufficient to **recover** all the private keys of a wallet
    - Typically backed up on paper
    - Optionally encrypted with password

Seed Example:

deal smooth awful edit virtual monitor term sign start home shrimp wrestle

Wallet classification



Hot and cold wallets

- Keys on an Internet-connected computer: **Hot** wallet
  - Easy to use
  - Can always spend my money immediately
- Private keys offline: **Cold** wallet
  - Kept on a computer not connected to the Internet or a hard drive
  - Keys cannot easily be stolen
  - Keys can be moved to a hot wallet when needed to spend
  - User can see balance and how much money they have using public keys kept (safely) online

Other ways to store cold wallets

Paper wallet

- Private key is printed on a piece of paper
- Can optionally be encrypted with a secret password (which is remembered)

Brain wallet

- Private key is SHA256(“my dog’s name is Barbie”) or some other passphrase
- Full private key can be recovered by memory
- Extremely unsafe!
    - Many £££ stolen due to low entropy passwords

Hardware wallets

- Special hardware device used to store private keys
    - Most popular ones: Trezor, Ledger
- Cold wallet
- Connects to a computer via USB
- Keys never leave the device
- Device signs transaction and sends it to computer
- When transacting, addresses are verified by looking at a screen
- As hardware/software is specialized, much harder to “hack” or have bugs
- Works safely even if host computer is compromised
    - Host can censor transactions! (eclipse)
- Protected by a pin in case of theft
- Can be backed up into paper and/or other hardware wallets