Bitcoin mixnet

Bitcoin recipients can have perfect anonymity: they can create a new address at any time to receive a payment and there's nothing to connect that address to the recipient's identity or other Bitcoin addresses. Senders have to work harder to achieve that objective.

If we assume that participants can use existing anonymizing networks to exchange messages, we can design a mixnet to provide payment senders with the ability to send coins without identifying themselves.

Overview

A cluster of nodes. Each node is self-contained and independently operated and can serve as a simple single-stage coin exchange on its own. Each node maintains a database. The database maps local Bitcoin addresses (ie. addresses controlled by the node) to destination addresses. Mappings will be created as clients request transactions and deleted once transactions are complete, freeing the local addresses to be reused. Each node also has a pool of available coins from which it will fund deposits to destination addresses. The node cannot be used for transactions larger than the size of this pool.

Transaction Procedure

A transaction involving a single node looks like this:

1. The client connects via HTTP or some RPC mechanism to the node's published transaction address. This address is hosted on the Internet or some anonymizing network.
    a) The client requests a transaction from the node. The client must provide the destination Bitcoin address to which the server will deposit coins, and the transaction fee which the node will pay to the block generator when it sends the exchanged coins.
    b) The node replies with a Bitcoin address of its own, selected randomly from its pool of addresses not currently involved in a transaction.
    c) The node creates a record in its database mapping its Bitcoin address to the destination address.
2. The client sends coins to the node's Bitcoin address.
3. The node receives the coins and waits for confirmations as configured by the node operator.
4. The node subtracts a fee from the total coins it received as configured by the node operator and sends the remaining coins onward to the destination address. The node manipulates its wallet in such a way as to guarantee that the coins sent are taken from its existing pool of coins and that it does not retransmit the coins received in this transaction.
5. The node removes from its database the record mapping its local Bitcoin address to the destination address.

If no coins are received at the node after a timeout configured by the node operator, the node removes from its database the record mapping its local Bitcoin address to the destination address.

Directory Service

By a mechanism not specified in this document (but which could be a central directory server, DHT, or other) clients obtain a complete list of operational nodes, including the following information for each node:

- Transaction address
- Confirmations required
- Coin receipt timeout
- Maximum transaction size
- Fee

Chained Transactions

For greater deniability, clients can build chains of nodes, sending coins from one to another and having the coins exchanged each time.

1. The client chooses the nodes which will participate in a chained transaction and places them in order.
2. The client performs Transaction Step 1 on each node in the chain, beginning with the last node and ending with the first. The last node is given the final destination address as its destination address for the transaction. The second-last node is given the address provided by the last node, the third-last node is given the address provided by the second-last node, and so on.
3. The client then performs Transaction Step 2, sending coins to the address provided by the first node.
4. Each node performs Transaction Steps 3-5 as per usual. The nodes behave the same for a chained transaction as for a single-stage transaction.

Objections

1. It may be possible to examine the block chain and find the address of either the originator or final recipient of a transaction by looking for a time sequence of similarly sized transactions, if the counter-party address is already known.
2. Node operators can lie about their fees.
3. Node operators can steal the entire value of a transaction.
4. Node operators can retransmit the coins they received in a transaction, removing deniability from the client.
5. The final recipient of a payment can falsely claim not to have received a payment and claim that either the sender or a mixnet node operator is responsible. Likewise the originator of a payment can falsely claim to have sent a payment and claim that either the recipient or a mixnet node operator is responsible.

Counter-Objections

1. If a node operator fails to properly exchange coins, the evidence will be recorded in the block chain for the client to read. The operator of a node will lose reputation as a result.
2. If a node operator engages in theft it will not be possible to prove it by examining the block chain.
3. If the final recipient of a payment falsely claims not to have received payment, it will be possible to examine the block chain and show that a payment of the correct size was deposited at the recipient's address. It will not be possible to prove that the deposit in question was a result of the originator's transaction.
4. If an originator falsely claims to have sent a transaction it will not be possible to prove this by examining the block chain.

Possible extensions

1. During Transaction Step 1 prior to step 1a, the client sends the transaction size to the node. The node replies with its fee for that transaction size. If the client accepts, it proceeds with step 1a, otherwise it disconnects.

   This extension allows the client to send multiple payments from separate accounts, possibly separate wallets, to the node, which will wait for the entire transaction balance to accumulate in Transaction Step 3 before proceeding to Transaction Step 4. It also allows the node operator to charge a different fee depending on the size of the transaction or other factors.
2. During Transaction Step 1a, the client sends multiple destination addresses each with a transaction size and the transaction fee which the node will pay to the block generator when it sends the exchanged coins. The node will wait for the entire transaction balance to accumulate in Transaction Step 3 before proceeding to Transaction Step 4. In Transaction Step 4, separate payments will be sent to each destination in the sizes specified by the client during Transaction Step 1a.

   This extension allows the client to send payments to multiple destination addresses during the same transaction. One possible use for this functionality is to break a single payment into multiple 'streams' of different sizes as it passes through the mixnet, making it more difficult to attack as per Objection 1.
3. During Transaction Step 1a, the client specifies a time delay for each destination address. In Transaction Step 4, the node will wait for the specified delay to elapse before sending each payment to the destination address.

   This extension allows the client to add delays to each stage of a transaction, further complicating an attack as per Objection 1.