Peer-to-Peer Networking and Applications (2022) 15:290–303
https://doi.org/10.1007/s12083-021-01241-z

Clover: An anonymous transaction relay protocol for the bitcoin P2P network

Federico Franzoni[1] · **Vanesa Daza**[1]

Received: 18 December 2020 / Accepted: 30 August 2021 / Published online: 6 October 2021

Abstract
The Bitcoin P2P network currently represents a reference benchmark for modern cryptocurrencies. Its underlying protocol defines how transactions and blocks are distributed through all participating nodes. To protect user privacy, the identity of the node originating a message is kept hidden. However, an adversary observing the whole network can analyze the spread pattern of a transaction to trace it back to its source. This is possible thanks to the so-called *rumor centrality*, which is caused by the symmetry in the spreading of *gossip*-like protocols. Recent works try to address this issue by breaking the symmetry of the Diffusion protocol, currently used in Bitcoin, and leveraging proxied broadcast. Nonetheless, the complexity of their design can be a barrier to their adoption in real life. In this work, we propose Clover, a novel transaction relay protocol that protects the source of transaction messages with a simple, yet effective, design. Compared to previous solutions, our protocol does not require building propagation graphs, and reduces the ability of the adversary to gain precision by opening multiple connections towards the same node. Experimental results show that the deanonymization accuracy of an *eavesdropper* adversary against Clover is up to 10 times smaller compared to Diffusion.

Keywords Blockchain · Bitcoin · Anonymity · Data propagation

1 Introduction

Over the past few years, Bitcoin [1] has risen to an unprecedented level of popularity. Although many users still believe this system to be anonymous, studies showed how it is relatively easy to link transactions to real identities [2–4]. Most deanonymization techniques work by tracing transactions on the blockchain and combining them with publicly available knowledge [5–7].

However, a less-known approach is to link transaction messages to their originating node in the underlying P2P network [8–10]. This approach is based on the observation that the first device to broadcast a transaction in the network is likely the one that created it. To implement this approach, an adversary typically deploys one or more nodes connecting to all reachable peers in the network, and listens for incoming transaction messages [11]. Transactions are then linked to their source by using an estimation strategy. This type of adversary is known as the *eavesdropper* adversary.

A recent work by Fanti et al. [12] shows that the Diffusion protocol, currently used in Bitcoin, has poor anonymity guarantees against this adversary. In particular, an attacker can obtain high levels of precision even when controlling just few nodes in the network and using a naive estimation strategy. The authors identify the problem in the symmetry of the spreading pattern: since transactions spread from each node to all its peers, it is always possible to determine the approximate point in the network where the propagation started. This phenomenon is known as *rumor centrality* and is specific to all gossip-like systems [13, 14].

Following these findings, few solutions have been proposed that reduce the ability of the adversary to identify the source of a transaction [15, 16]. These proposals break the symmetry in the propagation pattern by having nodes delegate the broadcast of new transactions to other nodes of the network. In particular, transactions are first propagated (or *proxied*) linearly over a path of nodes, and then broadcast using the Diffusion protocol. The way nodes are

✉ Federico Franzoni
fed.franzoni@gmail.com

Vanesa Daza
vanesa.daza@upf.edu

[1] Department of Information and Communication Engineering, Universitat Pompeu Fabra, Roc Boronat, 138, Barcelona 08018, Spain

chosen during this initial phase is defined by the protocol, and determines the security and complexity of the solution. In Dandelion [15], reachable nodes in the network build a propagation graph passing through every node (i.e., an Hamiltonian Circuit) and always propagate new transaction over the same path. Given the risk of the adversary learning the topology of such graph, a new graph has to be built periodically. In [16], the initial phase alternates reachable and unreachable nodes with the goal of concealing the propagation process from the adversary. Since the adversary can control multiple unreachable peers for any given node, the authors suggest the use of *bucketing* [17] to mitigate her ability of tracking transactions.

While these solutions sensibly improve the anonymity properties of transaction propagation, their adoption is hindered by their complexity. Additionally, in both protocols, the adversary can gain an advantage by learning sensitive information on the initial phase, such as the nodes in the propagation path (in Dandelion) or the transactions being proxied (in [16]).

In this paper, we propose a new propagation protocol that breaks the symmetry by separating inbound and outbound connections in the relay pattern. Our design is simpler than previous solutions, making its analysis and implementation easier. Additionally, we minimize security concerns by limiting the ability of the adversary to learn sensitive information: our protocol does not require build a propagation graph, and prevents the adversary from tracking transactions by isolating inbound connections. We formally analyze our protocol and experimentally evaluate it by using a proof of concept in a simulated environment. Our results show that, compared to Diffusion, our protocol reduces the deanonymization precision for the eavesdropper adversary from 0.6 to just 0.05, in the best case, and from 0.7 to 0.3 in the worst case.

2 Background

2.1 The Bitcoin P2P network

The Bitcoin P2P network is composed of nodes randomly connected among each other. Peers of a node are distinguished between outbound, whose connection was opened by the node, and inbound, from which the node accepted an incoming connection. According to the Bitcoin reference client, each node establishes and maintains 8 outbound connections and, if reachable, up to 117 inbound connections. Thus, reachable nodes can have up to 125 connections, while unreachable nodes are limited to 8.

However, this limit is not enforced, making nodes able to establish as many connections as needed. This is particularly useful for measuring tools that connect to all reachable nodes, as well as for the so-called *supernodes*, which are often used by mining pools to maximize their connectivity with the network. At the same time, malicious actors can exploit this feature to improve the effectiveness of their attacks.

2.2 Transaction propagation

A transaction *tx* is transmitted from a node *A* to a node *B* following a three-step process:

1. Node *A* announces *tx* to node *B* by sending an `INV` message, containing the hash of the transaction (*h(tx)*);
2. If *h(tx)* is unknown, node *B* requests *tx* to node *A* by sending a `GETDATA` message is sent, containing *h(tx)*;
3. Node *A* sends *tx* to node *B* via a `TX` message.

This announcement-based propagation mechanism is used to avoid transmitting a transaction twice to the same node. To optimize network data consumption, `INV` messages usually aggregate multiple transaction hashes.

Transactions are spread from a node to its peers following a gossip-like protocol known as *Diffusion*, which works this way: when a new transaction is created or received by a peer, it is announced to all connected peers; before sending the `INV` message, an individual random delay is applied to each peer.

2.3 Deanonymization strategies

In Bitcoin, the broadcast and relay operations follow the same rules. Therefore, when nodes create a transaction, they propagate it the same way as transactions received from their peers. This approach is used to prevent leaking the identity of the node that originates the transaction. However, as we already mentioned, it is possible to determine the source of a transaction by observing its propagation through the network.

In particular, an eavesdropper adversary, which connects to all reachable nodes in the network, can adopt different strategies to estimate the source of a transaction. The simplest method, called *first-timestamp* or *first-spy* estimator, consists in linking each transaction to the first node that announces it (to the adversary). The rationale behind this method is that the first node to announce a transaction in the network is likely the one that generated it. By connecting to all nodes, the adversary is always likely to receive each transaction from its source. This strategy has been proved to reach very high levels of accuracy against Diffusion, even when the adversary controls only few nodes [12]

More advanced techniques are theoretically possible when the adversary knows the network topology [12]. These techniques take into account the propagation of transactions

to exploit the rumor-centrality property of the Diffusion protocol. In particular, these techniques are based on the observed order in which nodes announce the transaction. The underlying assumption is that nodes that are (topologically) closer to the source will announce the transaction earlier than those that are farther away. Although these methods are theoretically more precise than the first-spy estimator, their adoption is conditioned to the knowledge of the network graph, which is currently obfuscated by the Bitcoin protocol.

3 Adversarial model

We consider the eavesdropper adversary defined in [12], which is based on practical attacks such as [8] and [9]. This adversary makes use of a supernode that connects to all reachable nodes in the network. For each node, multiple connections can be established by using different IP:port addresses, making them look as coming from different entities. In particular, the adversary can fill up all unused inbound slots of a target node.

Additionally, we extend this adversary by letting it deploy an arbitrary number of reachable nodes with the objective of being selected as an outbound peer by other nodes. This extension allows the adversary to improve precision against our protocol.

The goal of the adversary is to determine the source node for all received transactions. To this purpose, adversarial nodes listen to all messages relayed by their peers, logging their content and timestamp. We assume the adversary has a unified view of the logs from all the nodes under its control. Furthermore, for each honest node, the adversary maintains a *deanonymization set*, which contains all the transactions that have been possibly generated by that node. To deanonymize transactions, the adversary adopts the first-spy estimator: each transaction is linked to the first node that announces it (to any of the nodes controlled by the adversary).

4 The clover protocol

In this section, we describe Clover, our new transaction propagation protocol. We detail and motivate its design with reference to the adversary model.

4.1 Protocol overview

Similar to previous solutions [15, 16], Clover protects the source of a transaction by means of *proxying*. This consists in delegating the broadcast of new transactions to other nodes. Specifically, when a node creates a new transaction, it selects one of its peers and sends it the transaction. The selected node, called *proxy*, is then responsible for broadcasting the transaction to the rest of the network. Proxying allows moving the apparent origin of the propagation of a transaction from its source to a different node of the network.

Note that proxying drastically reduces the effectiveness of the first-spy estimator approach, since it is highly unlikely for the eavesdropper adversary to receive a transaction from its source. However, if the adversary controls the selected proxy, she could be able to distinguish a proxied transaction and simply link it to the sender node (i.e., deanonymize it).

To mitigate this risk, we use transaction *mixing*. This consists in making nodes proxy their new transactions along with transactions created (and proxied) by other nodes. This strategy reduces the ability of an adversarial node of determining whether a proxied transaction was created by the sender or a different node. In particular, the more the transactions used for mixing, the lower the precision of the adversary. We call *mixing set* the set of transactions used by a node for mixing.

To enable mixing, new transactions are proxied over multiple nodes before being broadcast (multi-hop proxying). In other words, a new transaction propagates in two phases: the *proxying phase*, in which the transaction is relayed through a number of proxy nodes, and the *diffusing phase*, where the transaction is broadcast and propagated following the Diffusion protocol. The switch between the proxying phase and the diffusing phase can occur at any hop, and it is determined probabilistically by the node that receives it. Specifically, when a node receives a transaction in the proxying phase, it decides whether to relay it to another proxy (thus keeping it in the proxying phase) or broadcast it with Diffusion (thus switching to the diffusing phase). We call *proxy transaction* a transaction in the proxying phase, and *diffused transaction* a transaction in the diffusing phase.

To improve anonymity, only proxy transactions are used for mixing. This is motivated by the observation that diffused transactions are likely to be known by the adversary. In fact, by connecting to all reachable nodes, an eavesdropper adversary is always among the first to receive such transactions. As such, the adversary is able to distinguish diffused transactions and exclude them from the deanonymization set, thus improving precision. This means that diffused transactions do not actually contribute to mixing. On the other hand, proxy transactions are likely unknown to the adversary and are thus ideal for mixing. In particular, when receiving a proxy transaction from a node, the adversary is not able to determine whether it was created by such node or by one of its peers. Hence, the anonymity of the mixing set solely depends on the number of proxy transactions it contains. Therefore, we maximize anonymity by having nodes include only proxy transactions in their mixing set.

In order to make nodes able to distinguish proxy transactions, we propagate them using a separate protocol

message, called PTX. Instead, diffused transactions will be transmitted using the standard TX message.

To further mitigate the risk of selecting an adversarial proxy, we only relay new transactions to outbound peers. This strategy is motivated by the fact that the adversary can control an arbitrary number of inbound connections. Instead, she has limited influence on the outbound peers, which are chosen at random among all reachable nodes in the network.

Following the same reasoning, we exclude from the mixing set the transactions received from inbound peers. We do this to limit the ability of the adversary to track transactions in the mixing set of a node. In fact, an adversary being used as a proxy by a node, and controlling many inbound connections towards the same node, can track all the transactions she relays through these connections and then exclude them from the corresponding deanonymization set to improve precision. At the same time, to allow a correct propagation, we have nodes relay proxy transactions from inbound peers to other inbound peers.

In summary, proxy transactions received from an outbound peer are relayed to another outbound peer, while proxy transactions received from an inbound peer are relayed to another inbound peer. We depict this scheme in Fig. 1. The name Clover has been chosen to recall the four-way pattern of our relay protocol.

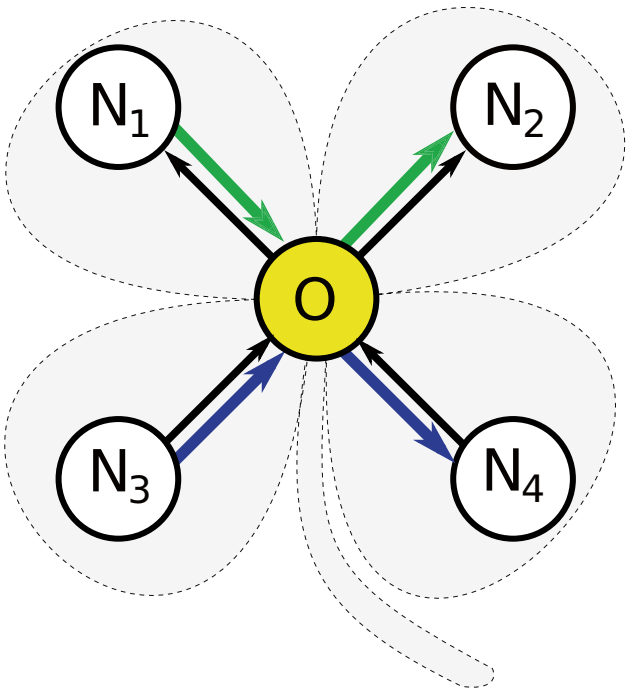

Fig. 1 The Clover relay protocol pattern: black arrows represent the direction of the connection; colored arrows represent relays of proxy transactions

4.2 Protocol design

In this section, we detail the Clover protocol design and describe the rules followed by network nodes.

4.2.1 Proxy transactions

We introduce a new protocol message PTX, used to propagate proxy transactions. The PTX message has the same structure as TX and is only used to mark a transaction in the proxying phase. Like the TX message, the PTX message contains the full transaction data.

During the proxying phase, transactions are propagated directly from one node to another, without previously announcing them via INV messages. In fact, the standard three-step transmission is meant to avoid sending a transaction twice to the same node, which is likely to occur in gossip-like protocols. However, since proxy transactions are propagated over a linear path, nodes are rarely expected to receive them twice. Instead, the receiver of a proxy transaction is always expected not to know it.

An upside of this strategy is that it allows us to substantially reduce the propagation delay introduced by the proxying phase. Specifically, since each relay operation only requires one message instead of three, the delay is reduced by one third.

4.2.2 Transaction propagation

When a node creates a new transaction *tx*, it selects a random proxy among its outbound peers, and sends it *tx* using a PTX message. This marks the beginning of the proxying phase for *tx*.

During this phase, at each hop, the transaction is relayed (re-proxied) to another node, or broadcast via Diffusion. A node *N* can receive a proxy transaction *tx* from both outbound and inbound peers. When this occurs, *N* behaves like follows: if *tx* is received from an outbound peer, *N* relays it to another outbound peer, chosen at random; if *tx* is received from an inbound peer, *N* broadcasts it with probability p, or relays it (with probability $1-p$) to an inbound peer, chosen at random. The probability p is defined at a global level, and determines the average number of hops through which a transaction is relayed during the proxying phase. When a transaction gets broadcast, it enters the diffusing phase and follows the standard Diffusion protocol.

Note that the broadcast step can only occur when *tx* is received from inbound peers. In other words, proxy transactions received from outbound peers are always re-proxied. This choice allows nodes to maximize their mixing set by

using all suitable transactions (i.e., those received from outbound peers).

4.2.3 Timeout

As the diffusion step is probabilistic, a transaction could be relayed too many times, producing an excessive delay in its propagation.

To mitigate this risk, when a node proxies a transaction, it sets a timeout to verify that it gets correctly diffused. To do so, the node monitors `INV` messages coming from their outbound peers. When the timeout expires, the node checks if the majority of the outbound peers has advertised the transaction. If so, the transaction is considered as correctly diffused; otherwise, the node broadcasts the transaction using Diffusion.

Again, we only consider outbound peers because they are the least likely to be controlled by the adversary. If we relied on inbound peers, an adversary, controlling the selected proxy and the majority of inbound peers, could trick the node by simply advertising the transaction from all such peers.

Note that the timeout is applied to all proxied transactions, regardless of being new or relayed. This prevents the adversary from distinguishing the two cases, which could lead to deanonymization attacks.

A default value for the timeout can be defined after performing experiments on the network. However, each node might choose its own value, depending on the desired security level.

4.2.4 Clover procedures

We first define the *proxy* procedure as in Algorithm 1.

Algorithm 1 Proxy procedure

```
   ENV: Timeout t; NodeSet OutPeers
1: procedure PROXY(Transaction tx, NodeSet ProxySet)
2:     ProxySet := ProxySet \ {tx.source}
3:     proxy = pickRandomNode(ProxySet)
4:     Send PTX(tx) to proxy
5:     wait(t)
6:     confirmations := Count( INV(tx) received from OutPeers )
7:     if confirmations < (|OutPeers|/2 + 1) then
8:         DIFFUSE(tx)
```

This procedure takes as inputs the transaction to be proxied (*tx*) and a set of peers (*ProxySet*) among which to choose the proxy. The procedure picks a random node from *ProxySet* and sends it a `PTX` message containing *tx*. If the transaction is being relayed, its sender is excluded from the candidates (to avoid sending the message back to the sender). After sending the `PTX` message, a timeout *t* is set. While *t* is not expired, the node collects `INV` messages from its outbound peers. When *t* expires, the node checks if the majority of outbound peers has announced *tx*. If so, the transaction is considered as diffused; otherwise, the transaction is broadcast.

We then define the Clover propagation rules as in Algorithm 2.

Algorithm 2 Clover Propagation Rules

```
1: ENV: Probability p; NodeSet OutPeers, InPeers
2: procedure CLOVER
3:     if New(Transaction tx) then
4:         PROXY(tx, OutPeers)
5:     if Receive(PTX(tx)) then
6:         if tx.source in OutPeers then
7:             PROXY(tx, OutPeers)
8:         else
9:             d = getRandProb()
10:            if d < p then
11:                DIFFUSE(tx)
12:            else
13:                PROXY(tx, InPeers)
```

When a node creates a new transaction *tx*, or receives `PTX`(*tx*) from an outbound peer, it runs *Proxy*(*tx*, *OutPeers*); if a `PTX`(*tx*) message is received from an inbound peer, the node runs *Diffuse*(*tx*) with probability *p*, and *Proxy*(*tx*, *InPeers*) with probability 1−*p*.

5 Discussion

In this section, we study the anonymity properties of the Clover protocol against an eavesdropper adversary using the first-spy estimator.

Notation. We use *R* to denote the set of reachable nodes in the network and *S* to denote the subset of reachable nodes controlled by the adversary (spies). Without loss of generality, we let *I* and *O* represent the average set of inbound and outbound peers of a node in the network.

We use the term *source* or *origin* of a transaction to indicate the node that created it. Instead, we use the term *sender* to indicate the node that sends a specific message.

For the sake of readability, Table 1 summarizes all parameters used in this section.

5.1 Security

We consider an eavesdropper adversary $\mathcal{A}$ as described in Sect. 3. As we will show, $\mathcal{A}$ gains no advantage by connecting to all nodes, nor by establishing multiple connections towards the same node. In fact, in our protocol, new transactions are only relayed through outbound connections, making the inbound peers controlled by $\mathcal{A}$ irrelevant to deanonymization. Instead, $\mathcal{A}$ gains precision by deploying more reachable nodes, as this increases her chances of being selected as a proxy node for new transactions.

To analyze the anonymity properties of Clover, two important aspects must be studied first. On the one hand, we need to know the probability of selecting an adversarial node as proxy for new transactions. On the other hand, we

Table 1 Parameter definitions

Parameter	Description
$\mathcal{A}$	The eavesdropper adversary
R	Set of reachable nodes
O	Average set of outbound peers of a node
I	Average set of inbound peers of a node
p	Probability of diffusion
S	Set of nodes in R controlled by $\mathcal{A}$
ρ_I	Average number of transactions received from an inbound peer
σ_I	Average number of transactions sent to an inbound peer
ρ_O	Average number of transactions received from an outbound peer
σ_O	Average number of transactions sent to an outbound peer
$\mathcal{M}$	Average mixing set of a node
a	Adversarial outbound peers of a node
g	Average number of transactions generated by a node

need to determine the size of the average mixing set for a single node.

With these values, we can calculate the precision of $\mathcal{A}$ in deanonymizing proxy transactions, as well as its overall precision against all new transactions. Note that $\mathcal{A}$ will mainly target proxy transactions, as other transactions are unlikely to be announced by their source, in the Clover protocol.

Proxy Selection For the sake of simplicity, we assume each reachable node has the same probability of being selected as outbound peer when a new node joins the network[1].

Thus, we compute the probability of selecting an adversarial proxy for a single transaction as follows:

Lemma 1 *Let R be the set of reachable nodes, and S be the subset of nodes in R controlled by the eavesdropper adversary $\mathcal{A}$, then the probability $P_{\mathcal{A}}$ of selecting an adversarial node as the proxy for a new transaction is*:

$$P_{\mathcal{A}} = \frac{|S|}{|R|}. \tag{1}$$

Proof As each node establishes $|O|$ outbound connections, the probability of selecting a node in R as an outbound peer is $|O|/|R|$. As the adversary controls $|S|$ nodes in R, the probability of selecting a node in S as an outbound peer is $|O||S|/|R|$. Since new transactions are sent to a random node in O, the probability of selecting a node in S for a single new transaction is:

$$P_{\mathcal{A}} = \frac{1}{|O|} \cdot \frac{|O||S|}{|R|} = \frac{|S|}{|R|}. \tag{2}$$

□

Therefore, in the current Bitcoin network, where $|R| \approx 10,000$, $\mathcal{A}$ would have $1/10000 = 0.0001$ probability of being selected as a proxy when controlling a single node. On the other hand, when controlling 1000 nodes (10% of the reachable network) $\mathcal{A}$ would have 0.1 probability of being selected for each new transaction sent in the network.

Note that we are not taking into account other protective measures used by the Bitcoin client, such as the limitation in the number of peers from a single subnet, or the use of bucketing [17]. Since such measures are explicitly meant to reduce the probability of connecting to multiple adversarial nodes, it is likely that including these factors in the analysis would lower the value of $P_{\mathcal{A}}$.

5.1.1 Transaction mixing

To ease the analysis, we study the mixing property of a node over a period of time T. However, as we will see, results are independent from this value.

We want to calculate the average size of the mixing set of a node, which corresponds to the number of `PTX` messages received from outbound peers (i.e., nodes in O). In the following, we will use the word *transaction* as a synonym of `PTX` message.

We use ρ_I and σ_I to denote the average number of transactions received from and sent to each node in I, respectively. Similarly, we use ρ_O and σ_O for transactions received from and sent to nodes in O.

We study the size of the average mixing set $\mathcal{M}$ for a node having a adversarial outbound peers. Note that, when all outbound peers are honest, the mixing set

[1] Although this assumption is theoretically sound, in the real Bitcoin network, well-established nodes tend to have more connections, especially compared to newly-joined nodes. This fact lowers the probability of connecting to the adversary, unless she is in control of a large portion of well-established nodes.

contains all transactions received from such peers (i.e., $|\mathcal{M}| = \rho_O|O|$). However, if $\mathcal{A}$ controls one or more outbound peers, the transactions received from these nodes are not useful for mixing (since they are known to $\mathcal{A}$). Therefore, in this case, the size of the average mixing set is $|\mathcal{M}| = \rho_O(|O| - a)$.

Given the above, the following equation holds:

Lemma 2 *Let n be a generic node of the network, g be the average number of transactions generated by n, O be the set of outbound peers of n, a be the subset of O controlled by $\mathcal{A}$, and p be the probability of Diffusion in Algorithm 1. Then, the cardinality of the mixing set $\mathcal{M}$ for a node n is*:

$$|\mathcal{M}| = \frac{g(1-p)}{p} \cdot \frac{|O| - a}{|O|}. \tag{3}$$

Proof We consider the mixing set in the presence of a adversarial nodes among the outbound peers: $|\mathcal{M}| = \rho_O(|O| - a)$. By definition, $\rho_O = \sigma_I$. Given the rules defined in Algorithm 2, transactions received from nodes in I (ρ_I) are relayed, with probability $1 - p$, uniformly at random among nodes in I. Thus, we have:

$$\sigma_I = (\rho_I|I|(1-p))/|I| = \rho_I(1-p). \tag{4}$$

By definition, $\rho_I = \sigma_O$. Let us assume each node generates an average of g transactions during T. Given that each node sends to nodes in O all of its transactions along with those received by other nodes in O, we have: $\sigma_O = (g + \rho_O|O|)/|O|$.

Given that $\rho_O = \sigma_I$ and $\rho_I = \sigma_O$, we have:

$$\sigma_O = \frac{(g + \sigma_O(1-p)|O|}{|O|}. \tag{5}$$

Isolating σ_O, we get

$$\sigma_O = \frac{g}{|O|p}. \tag{6}$$

On the other hand, as $\rho_O = \sigma_I = \rho_I(1-p) = \sigma_O(1-p)$, we obtain:

$$\begin{aligned} |\mathcal{M}| &= \rho_O(|O| - a) \\ &= \sigma_O(1-p)(|O| - a) \\ &= \frac{g}{|O|p}(1-p)(|O| - a) \\ &= \frac{g(1-p)}{p} \cdot \frac{|O| - a}{|O|}. \end{aligned} \tag{7}$$

□

Note that the size of the mixing set is inversely proportional to p. In fact, the smaller this value, the longer a transaction will be relayed before being diffused. In turn, the more a transaction is relayed, the more it contributes to the mixing of the other nodes.

5.1.2 Deanonymization precision

As previously mentioned, we expect $\mathcal{A}$ to mainly target proxy transactions, since it will be highly unlikely for her to receive diffused transactions from their source. Therefore, we first study the precision of $\mathcal{A}$ against the proxy transactions she receives. Then, we compute the overall accuracy considering all transactions.

First, let us consider the precision against proxy transactions coming from a single node. Note that this only applies to nodes that opened a connection towards an adversarial peer. We assume $\mathcal{A}$ does not know incoming proxy transactions (although this might occasionally happen). As the first-spy estimator is used, each transaction is linked to the node that relayed it.

Let D_{proxy} be the average precision of $\mathcal{A}$ against proxy transactions coming from a single node. Then:

Lemma 3 *Let n be a generic node of the network, O be the set of its outbound peers, a be number of peers in O controlled by the eavesdropper adversary $\mathcal{A}$, and p be the probability of Diffusion in Algorithm 1. Then, the average precision of $\mathcal{A}$ against proxy transactions from a node n is*:

$$D_{proxy} = \frac{p}{1 - \frac{a(1-p)}{|O|}}. \tag{8}$$

Proof We consider a node n generating g transactions, and being connected to a outbound peers controlled by $\mathcal{A}$. As both new and relayed transactions are distributed among nodes in O, each such node receives on average $g/|O|$ new transactions plus $|\mathcal{M}|/|O|$ mixing transactions. Since $\mathcal{A}$ associates all transactions to n, she will get $g/|O|$ correct guesses over $(g + |\mathcal{M}|)$ transactions received.

By Lemma 2, we get:

$$\begin{aligned} D_{proxy} &= (g/|O|)/((g + |\mathcal{M}|)/|O|) \\ &= g/(g + |\mathcal{M}|) \\ &= g/(g + g\frac{1-p}{p}\frac{|O| - a}{|O|}) \\ &= \frac{p}{1 - \frac{a(1-p)}{|O|}}. \end{aligned} \tag{9}$$

□

To calculate the overall precision, we consider a network of $|N|$ nodes, $|R|$ of which are reachable. Let $D_{overall}$ be the overall precision of $\mathcal{A}$ against transactions generated by nodes in N. Then, the following equation holds:

Lemma 4 *Let R be the set of reachable nodes, and S be subset of adversarial nodes in R. Then, the overall average precision of the eavesdropper adversary $\mathcal{A}$ against new transactions in the network is*:

$$D_{overall} = \frac{|S|}{|R|}. \tag{10}$$

Proof Let us consider all transactions generated by nodes in N, that is gN. By Lemma 1, each transaction is sent to an adversarial proxy with probability $|S|/|R|$. As such, $\mathcal{A}$ will receive $gN(|S|/|R|)$ transactions from their source (thus guessing them correctly). Dividing correct guesses over the total amount of transactions we have:

$$\frac{N \cdot \frac{|S|}{|R|} g}{gN} = \frac{|S|}{|R|}. \tag{11}$$

□

Therefore, the overall precision only depends on the portion of reachable nodes controlled by $\mathcal{A}$.

5.2 Complexity and efficiency

As it can be seen by the Clover procedures (Algorithms 1 and 2), the algorithm followed by network nodes has only plain instructions and if/then statements, without any loop. Since its complexity is constant (O(1)), Clover does not add any computational overhead to the Bitcoin protocol.

Similarly, there is no expected overhead in the number of exchanged messages. In fact, like in the Diffusion protocol, transactions are propagated through all nodes of the network, without repetitions (although this can occasionally occur in Clover). Instead, since transactions in the proxying phase are transmitted directly (without previously announcing them), the total number of messages exchanged per node is expected to be lower than Diffusion.

On the other hand, like other similar solutions, the Clover protocol introduces a delay in the broadcast of a transaction. Specifically, this delay is proportional to the number of hops through which transactions go during the proxying phase.

In this respect, two factors must be considered: the number of messages needed for each hop, and the number of hops.

5.2.1 Hop delay

As described in Sect. 2, in the Bitcoin protocol, each transaction propagation hop requires three messages: `INV`, `GETDATA`, and `TX`. This strategy is used in Diffusion to avoid sending transaction data to nodes that already have it.

In Clover, this is not needed, since proxy transactions are normally unknown to the recipient. Instead, transaction data is transmitted directly with a single `PTX` message. Therefore, only one extra message is needed for each hop in the proxying phase.

5.2.2 Proxy hops

As previously stated, a higher number of relays during the proxying phase corresponds to a bigger mixing set for nodes in the network (and hence, better anonymity). Nevertheless, if this number is too high, it can cause an excessive propagation delay. Therefore, it is essential to choose a target value that seeks a compromise between efficiency and effectiveness.

Note that the average number of hops directly depends on the probability p. In particular, the lower this value, the higher the number of hops. Therefore, we can choose p to obtain a target number of hops (h).

In Sect. 7, we calculate the relation between p and h, and experimentally evaluate the optimal target number of hops.

6 Comparison with state-of-the-art solutions

We compare Clover with other known anonymity-preserving propagation protocols. To the best of our knowledge, the only similar solutions proposed to date are Dandelion [15] (extended with Dandelion++ [18]), and the one proposed by Franzoni and Daza [16].

We review the main differences with Clover, and compare their complexity, efficiency, and security.

Dandelion This protocol, proposed by Bojja Venkatakrishnan et al. in [15] and extended in [18], is the first solution to have tried protecting transaction anonymity by breaking the symmetricity of propagation. Similar to Clover, Dandelion consists of two phases: a first lineal relay phase, called *stem*, and a second broadcasting phase, called *fluff*, where transactions are propagated using Diffusion. Transactions in the stem phase are relayed according to a propagation graph (a circle in Dandelion, and a 2-regular graph in Dandelion++), which is built by participating nodes prior to run the protocol. To that purpose each node selects one or two possible proxies (depending on the protocol version) uniformly at random among their outbound peers. By using a limited set of proxies, Dandelion aims at maximizing the mixing property since all transactions are relayed through the same path.

Both in Dandelion and Clover, transactions are propagated only through outbound connections, minimizing the risk of proxying new transactions through adversarial nodes. However, Dandelion use transactions received from inbound peers for mixing, thus leaving space for the adversary to

improve precision by controlling a large portion of inbound connections. For instance, let us consider the case in which an adversarial node is selected as a proxy by a victim; when this occurs, the adversary can open as many inbound connections as possible towards the victim to improve her chances of being used as inbound peer in the propagation graph. Whenever the adversary controls both one or more inbound peers and one or more outbound peers of such graph, she will be able to track all transaction used for mixing, and thus easily detect those generated by the victim. In Clover, we prevent this risk by only mixing with transactions relayed by other outbound peers. In other words, the adversary cannot gain any precision by opening inbound connections towards a victim.

The use of a propagation graph in Dandelion not only increases the complexity of the protocol, but also introduces an additional attack vector for the adversary, which can gain precision by learning the topology of such graph. To avoid this risk, such graph has to be renewed every ten minutes, thus further increasing the complexity of the protocol. Clover avoid these issues by making use of all connected peers, and selecting proxies at random for each relay operation.

With respect to the delay introduced by the initial proxying phase, Clover also outperforms Dandelion by transmitting transactions directly, without using the three-step relay process described in Sect. 2.2. Roughly speaking, one hop in the stem phase of Dandelion introduces the delay of three hops in Clover. In other words, the delay introduced by each proxy hop in Clover is approximately one third than in Dandelion. Note that this allows for a longer proxying phase, which, in turn, means better anonymity properties (as the security of proxy transactions also depends on the average number of hops in such phase).

Finally, a major limitation of Dandelion is to be only compatible with reachable nodes, which represent only the 10% of the whole Bitcoin network. This is due to the fact that it requires nodes to have inbound connections. Conversely, Clover can also works when only outbound connections are available, thus being compatible with all nodes in the network.

Reachability-dependent Anonymous Propagtion (ReAP) Following an approach similar to Dandelion, Franzoni et al. [16] proposed an alternative protocol that breaks the symmetry of transaction propagation by leveraging unreachable nodes. We call this protocol *Reachability-dependent Anonymous Propagation*, or ReAP.

In ReAP, transactions are again propagated in two phases, the first one of which have them relayed linearly through a sequence of proxy nodes. In the initial phase transactions are relayed through an alternate sequence of reachable and unreachable nodes. In particular, reachable nodes proxy transactions via unreachable nodes, and viceversa. This strategy has a twofold goal: to improve the involvement of unreachable nodes in transaction propagation, and to limit the ability of the adversary to observe the propagation pattern through the network. Their approach is based on the observation that the adversary is unable to open connections towards unreachable nodes, and hence cannot observe propagation through such nodes.

Similar to Clover, ReAP does not require building a graph, is compatible with unreachable nodes, and minimizes delay by relaying transactions directly (i.e., without first announcing them) during the proxying phase. However, there are two major flaws in this protocol. First of all, reachable nodes require unreachable nodes to be connected in order to implement the protocol, which is not always the case. For instance, newly-joined nodes will likely have no inbound peers until their address is advertised to enough peers. Clover has no such limitation and can be readily be used by any node as soon as they connect to the network.

The second major issue in ReAP lies in the ability of the adversary to open multiple connections from unreachable nodes towards reachable ones, which increases her chances to be selected as proxy for new transactions. Furthermore, this allows her to track a many transactions in the mixing set of the target, which, in turn, helps her improve precision in deanonymization. In Clover, we prevent this issue by having nodes mix only with transactions from other outbound peers, thus minimizing the ability of the adversary to track transactions in the mixing set of a target.

Finally, the ReAP design has to deal with the complexity of determine the reachability of each node. In fact, this information is not explicit in the protocol and can only be inferred by probing the public listening address of a node. However, such address is not always advertised by nodes, making it hard to establish whether an inbound peer is reachable or not. Clover avoids such complexity by only differentiating between outbound and inbound connections, whose difference is well defined by the Bitcoin protocol and can be easily verified at any time.

Finally, different from ReAP, we prove the anonymity guarantees of our protocol, both by formal analysis and experimental results.

Summmary In the previous paragraphs, we compared Clover with state-of-the-art anonymous transaction propagation protocols. In particular, we discussed the complexity of their design, their scope, their security against the eavesdropper adversary, and the overhead introduced by the anonymity phase.

In Table 2, we summarize this comparison. As explained, both Dandelion/Dandelion++ and ReAP requires extra computation in order to enable the protocol; in contrast, Clover can be used without any previous operation. As for the delay introduced by the proxying phase, we saw how Clover and ReAP both minimize it to one extra message per hop. For what

Table 2 Comparison between Clover, Dandelion, and ReaP

Protocol	Extra Computation	Overhead	Scope	Extra Precision
Dandelion [15] **Dandelion++** [18]	Yes	3 msg/hop	Reachable only	Yes
ReAP [16]	Yes	1 msg/hop	All except new reachable nodes	Yes
Clover	No	1 msg/hop	All	No

concerns the scope of the protocol, Clover is the only one that can be used by all nodes in the network, since Dandelion/Dandelion++ is only compatible with reachable nodes, while ReAP cannot be used by newly-joining reachable nodes. Finally, both Dandelion/Dandelion++ and ReAP allow the adversary to gain extra precision in deanonymization by means of side channels: in the first case, by learning the topology of the graph, and in the latter case, by connecting to a reachable node with multiple unreachable peers. On the contrary, an eavesdropper adversary can only gain precision against Clover by controlling a larger portion of reachable nodes.

In summary, Clover combines the strengths of previous solutions while mitigating their limitations and security risks. The resulting protocol has strong anonymity guarantees for all nodes in the network, with a simple design and minimum overhead.

7 Experimental results

To evaluate the effectiveness of our protocol against an eavesdropper adversary, we performed a series of experiments in a simulated environment. In each series, we varied the portion of the network controlled by the adversary, so as to study the resilience of the protocol.

We compare our results with those obtained using Diffusion in the same simulation setting. Our results show that Clover reduces the precision of the first-spy estimator up to ten times in the best case, while significantly increasing the cost of the attack for the adversary.

7.1 Proof of concept and simulation

We set our experiments in a private Bitcoin network using the reference client (Bitcoin Core 0.20), which we modified to implement the Clover protocol. For the experiments running the Diffusion protocol, we used the original implementation without modifications.

Setting To run the simulations we executed nodes in Regtest (private) mode using Docker containers. Each test was run on a network of 100 reachable nodes randomly connected to each other. In each simulation, we had nodes randomly generate transactions during 10 minutes. On average, in each run we generated approximately 300 transactions, with an average of 3 transactions per node. For each setting, we run 3 simulations and then computed the average.

Being a simulated environment, our experimental setting might not fully represent the actual Bitcoin network. In particular, unlike our simulation, connections in the real network are not evenly distributed among nodes. Instead, stable nodes often maintain more connections than others, while newly-joined nodes typically require several hours before having a stable number of inbound peers. This might be an advantage for the adversary in the case it runs a well-known, stable node. However, in the real network, the adversary will be likely more limited than in our simulation, due to the fact that she needs to deploy several nodes to perform the deanonymization attack. In other words, deanonymization results in our simulation are likely to be better than they would be in the real world.

Unlike the real Bitcoin network, our simulation maintains a stable topology during the experiments. The stability of a controlled environment allows us to better evaluate the effectiveness of our propagation protocol against deanonymization attacks. Moreover, it allows a more meaningful comparison between Clover and Diffusion, since they can be tested in similar conditions, without depending on the randomness of the real network.

Due to technical reasons, we also exclude unreachable nodes in our simulations. Note that this has no relevance for Clover, since we showed in Sect. 5 how precision exclusively depends on reachable nodes, but it might slightly affect the results for the Diffusion protocol. However, although unreachable nodes are theoretically relevant in Diffusion, studies showed how their involvement in the transaction propagation is extremely low compared to their number [19], with as little as the 0.001% of nodes sending transaction messages. We then consider this as a minor limitation.

Overall, despite the differences between our simulated environment and the actual Bitcoin network, we believe our results are proper indicator of the security gains of Clover over Diffusion.

Adversary We varied the number of adversarial nodes from 1 to 30, corresponding to a range between 1% and 30% of the reachable network. These nodes are chosen randomly among those already deployed, so that they are well connected to the rest of the network. Note that this is the worst-case scenario since it assumes the adversary controls well-established nodes in the network. In addition, when testing against Diffusion, each adversarial node connects to all reachable peers (recall that this is not necessary when testing Clover, since it does not improve the precision of the adversary).

All adversarial nodes log incoming `INV` and `PTX` messages. At the end of the simulation, these logs are merged and ordered by timestamp. Then, the first-spy estimator is applied, linking each transaction to the first peer that advertised or transmitted it to any of the adversarial nodes.

Timeout The diffusion timeout has been set to fit the local simulation environment, where transactions are produced and spread faster than the real network. In particular, verification timeout has been set to 1 minute.

7.2 Simulation results

We evaluated precision against adversaries controlling 1%, 2%, 5%, 10%, 20%, and 30% of the network. Each adversary is first tested against Diffusion, and then against Clover with broadcast probability p equal to 0.2, 0.3, and 0.4. Overall precision is calculated as the average among all tests with a given adversarial power. Results are shown in Fig. 2.

In our simulations, the precision of the adversary against Diffusion showed to be very high even controlling a small portion of the network. In particular, when controlling from 1% to 5% of the reachable network, the adversary had a precision as high as 0.6. This value raises to 0.7 when the number of adversarial nodes reaches the 20% of the network.

On the other side, precision against Clover, although growing faster in the number of adversarial nodes, showed to be much lower than against Diffusion. Specifically, overall precision is from 10 times smaller (0.05), for adversaries controlling from 1% to 5% of the network, to 3 times smaller (0.33), for adversaries controlling from 10% to 30% of the network.

For what concerns precision against proxy transactions, we have, as expected, better results for lower values of p. In particular, with $p = 0.3$, precision ranges from 0.16 to 0.4, while, for $p = 0.2$, the adversary showed an average precision of 0.14 when the controlling 1-5% of the reachable nodes, and up to 0.35 when controlling 30%. When setting $p = 0.1$ precision against proxy transactions gets as low as the overall precision, indicating a near-optimum level of mixing.

Notably, the precision of the adversary against Clover never exceeded that against Diffusion. This means that Clover against a strong adversary controlling 30% of the network outperforms Diffusion against the weakest adversary controlling 1% of the network.

A major result of our experiments is that it shows how attacking Clover is substantially more expensive for the adversary (who need to deploy numerous nodes), compared to Diffusion, without even reaching the same levels of accuracy.

Hops According to our experiments, the average number of hops is inversely proportional to the probability p. In particular, we found the following relation to hold:

$$h \approx \frac{(1-p)}{0.15}. \tag{12}$$

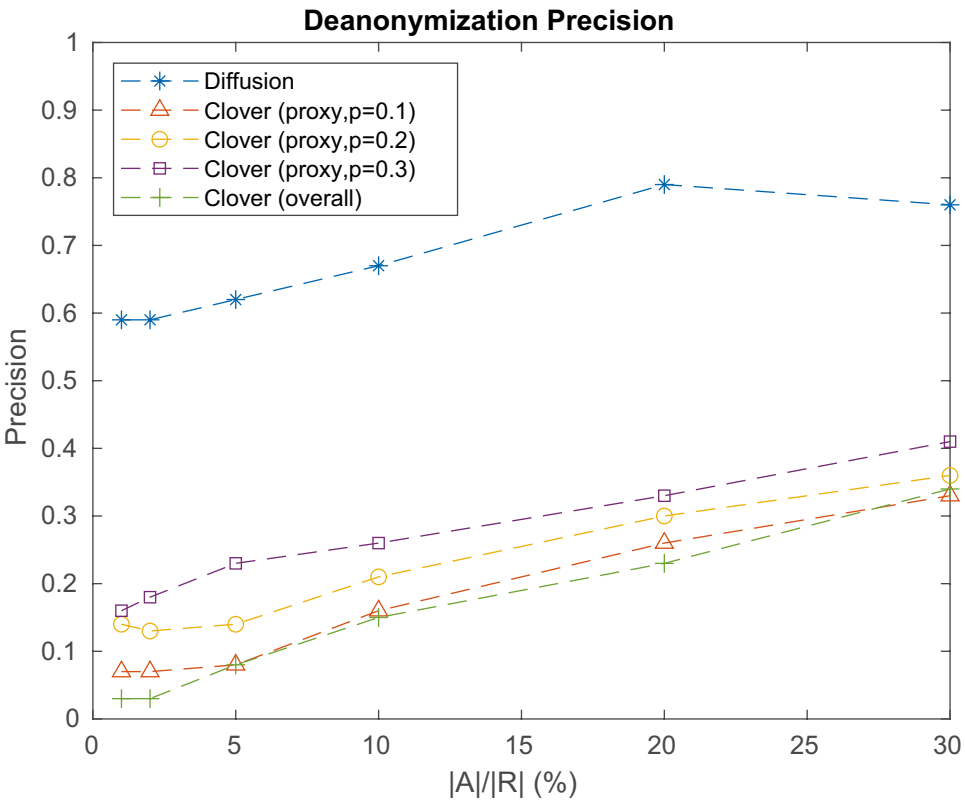

Fig. 2 Deanonymization precision against Clover

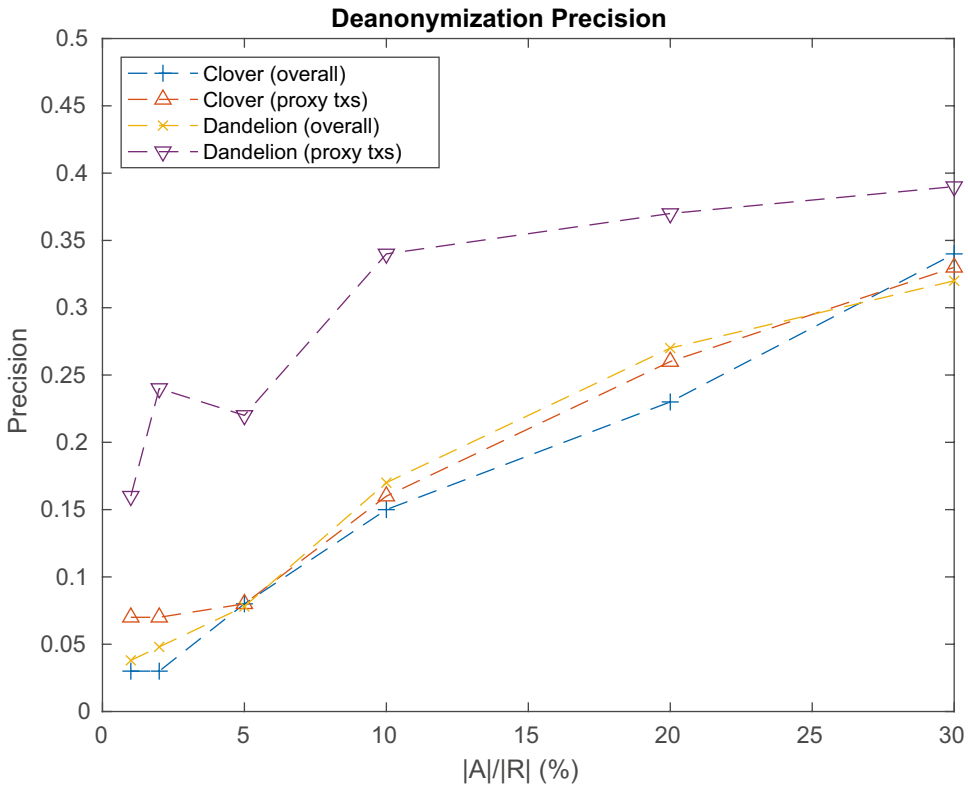

Fig. 3 Precision against Clover (p=0.1) and Dandelion

For instance, with probability $p = 0.1$, transactions are relayed through an average of 6 hops, during the proxying phase.

7.3 Comparison with dandelion++

To further demonstrate the benefits of Clover, we experimentally compared its results against Dandelion++. To that purpose, we run the same set of experiments, in the same setting, using the official implementation of Dandelion++. We compare its results against Clover when the probability of diffusion is set to $p = 0.1$, since this is also the value used by Dandelion++. Results are shown in Fig. 3.

As shown in the graphic, the overall precision of the eavesdropper adversary is comparable between Clover and Dandelion++. In contrast, precision against transactions in the proxying phase is visibly higher in Dandelion++. This is probably due to the fact that Clover better utilizes available transactions for the mixing property. In particular, while Clover distributes all incoming proxy transactions among all proxy nodes, Dandelion++ links each outbound proxy to a specific inbound peer. Additionally, all new transactions are proxied through the same node (during one epoch).

For the same reason, the results we obtained for Dandelion++ against proxy transactions were highly variable within a single setting. The irregularity of the corresponding line of the graphic reflects this variability.

In summary, when compared to Dandelion++, Clover shows a similar level of anonymity, but with better, and more stable, results for transactions in the proxying phase.

8 Related work

Anonymity in Bitcoin has been widely addressed in research [4]. In particular, two major directions have been explored in relation to deanonymization.

On the one side, there is blockchain analysis [3, 5], which aims at linking Bitcoin addresses (and all related transactions) to real-world identities. This is done by crossing publicly available information (e.g., known addresses or transactions, known services, ...) with address clustering: since all transactions are linked to each other, it is possible to trace coins throughout the whole blockchain. To prevent this kind of attack, users can use *mix services*, which allow them to shuffle their coins with other users so as to prevent the possibility of tracing back coins in the blockchain.

On the other side, there is traffic analysis, which aims at linking transactions to the IP address from which it originated, which would likely reveal the owner of the coins spent by such transactions. This is typically done by connecting to the whole network and monitoring transaction messages. Note that these attacks can go beyond the capacity of blockchain analysis, since they do not take into account Bitcoin addresses but only network

packages. In other words, network analysis can deanonymize a transaction even when this is anonymized through a mix service. Although anonymity networks, like Tor or I2P, can be used to protect from such attacks, these services are not commonly used by Bitcoin users, and might even lead to other deanonymization attacks [10]. Therefore, network-level anonymity is still a major concern for Bitcoin users. In the following, we review the most relevant works related to network-level deanonymization.

Network-Level Deanonymization Kaminsky [20] is the first one to propose the general first-spy approach. Based on the observation that nodes announce their transactions to all peers, he proposes to connect to all nodes and simply associates each transaction to the first node that announces it.

Koshy et al. [9] are among the first ones to apply the first-spy approach on the Bitcoin network. In their experiment, they connect to all nodes during 5 months and analyze the relay patterns of each transaction. Their results showed an accuracy of around 20% using very conservative thresholds.

In [21], Biryukov et al. propose a novel deanonymization technique targeting different cryptocurrency networks based on propagation analysis. Their approach is based on rumor centrality. According to their estimates, this technique is feasible even for low-budget adversaries.

Differently from other works, which only apply to reachable nodes, Biryukov et al. [8, 10] specifically target unreachable nodes and nodes using Tor. Their approach is based on fingerprinting techniques and reaches accuracy levels between 11% and 60%, depending on the stealthiness of the attacker. Since their technique only works during a single session, [22] propose a complementary technique that allows identifying unreachable nodes over multiple sessions.

In [12] and [11], Fanti et al. theoretically analyze the anonymity properties of Trickle and Diffusion protocols against an eavesdropper adversary using first-spy and rumor-centrality-based estimators. Their results show that both protocols have poor anonymity guarantees and identify the symmetry of the propagation pattern as the core issue.

9 Conclusion

Transaction anonymity is considered an essential feature of cryptocurrencies. However, while great improvements have been made at the application level, the network level is still vulnerable to cheap and effective deanonymization attacks. Recent proposals have identified and addressed the issues in the propagation protocol that lead to such attacks. Nonetheless, the complexity of the proposed solutions might hinder their adoption in real networks.

In this paper, we proposed an alternative approach to transaction propagation for the Bitcoin network, which adopts a simple design that eases its analysis and implementation. We theoretically studied its anonymity guarantees against powerful adversaries and experimentally evaluated its effectiveness through simulations, comparing results with the protocol currently used in Bitcoin.

Our experimental results show that the deanonymization precision of the eavesdropper adversary adopting the first-spy estimator is up to 10 times smaller in the best case. We believe our solution can be easily adopted in real cryptocurrency networks and serve as a basis for future advances in the field.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

References

1. Nakamoto S (2008) Bitcoin: A peer-to-peer electronic cash system. Bitcoin. 4 https://bitcoin.org/bitcoin.pdf
2. Androulaki E, Karame GO, Roeschlin M, Scherer T, Capkun S (2013) Evaluating user privacy in bitcoin. In A.-R. Sadeghi, editor, Financial Cryptography and Data Security, pp 34–51, Berlin, Heidelberg. Springer Berlin Heidelberg. ISBN: 978-3-642-39884-1
3. Reid F, Harrigan M (2013) An Analysis of Anonymity in the Bitcoin System. Springer New York, pp 197–223. ISBN: 978-1-4614-4139-7 https://doi.org/10.1007/978-1-4614-4139-7_10
4. Herrera-Joancomartí J (2015) Research and challenges on bitcoin anonymity. In: Garcia-Alfaro J, Herrera-Joancomartí J, Lupu E, Posegga J, Aldini A, Martinelli F, Suri N (eds) Data Privacy Management. Autonomous Spontaneous Security, and Security Assurance. Springer International Publishing, Cham, pp 3–16. ISBN: 978-3-319-17016-9.
5. Meiklejohn S, Pomarole M, Jordan G, Levchenko K, McCoy D, Voelker GM, Savage S (2013) A fistful of bitcoins: Characterizing payments among men with no names. In Proceedings of the 2013 Conference on Internet Measurement Conference, IMC '13, pp 127–140, New York, NY, USA. ACM. ISBN: 978-1-4503-1953-9. https://doi.org/10.1145/2504730.2504747
6. Nick JD (2015) Data-driven de-anonymization in bitcoin. Master's thesis, ETH-Zürich
7. Neudecker T, Hartenstein H (2017) Could network information facilitate address clustering in bitcoin? In: Brenner M, Rohloff K, Bonneau J, Miller A, Ryan PY, Teague V, Bracciali A, Sala M, Pintore F, Jakobsson M (eds) Financial Cryptography and Data Security. pp. Springer International Publishing, Cham, pp 155–169. ISBN: 978-3-319-70278-0.
8. Biryukov A, Khovratovich D, Pustogarov I (2014) Deanonymisation of clients in bitcoin p2p network. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, CCS '14, pp 15–29, New York, NY, USA. ACM. ISBN: 978-1-4503-2957-6. https://doi.org/10.1145/2660267.2660379

9. Koshy P, Koshy D, McDaniel P (2014) An analysis of anonymity in bitcoin using p2p network traffic. In Financial Cryptography and Data Security, pp 469–485, Berlin, Heidelberg. Springer Berlin Heidelberg
10. Biryukov A, Pustogarov I (2015) Bitcoin over tor isn't a good idea. In 2015 IEEE Symposium on Security and Privacy. pp 122–134. https://doi.org/10.1109/SP.2015.15
11. Fanti G, Viswanath P (2017) Deanonymization in the bitcoin p2p network. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems30, pp 1364–1373. Curran Associates, Inc. https://papers.nips.cc/paper/6735-deanonymization-in-the-bitcoin-p2p-network.pdf
12. Fanti GC, Viswanath P (2017) Anonymity properties of the bitcoin P2P network. CoRR, abs/1703.08761
13. Shah D, Zaman T (2012) Rumor centrality: A universal source detector. In Proceedings of the 12th ACM SIGMETRICS / PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS'12, pp 199–210. New York, NY, USA. Association for Computing Machinery. https://doi.org/10.1145/2254756.2254782
14. Shah D, Zaman T (2011) Rumors in a network: Who's the culprit? IEEE Trans Inf Theory 57(8):5163–5181. https://doi.org/10.1109/TIT.2011.2158885
15. Bojja Venkatakrishnan S, Fanti G, Viswanath P (2017) Dandelion: Redesigning the bitcoin network for anonymity. Proc ACM Meas Anal Comput Syst 1(1):22:1–22:34. https://doi.org/10.1145/3084459
16. Franzoni F, Daza V (2020) Improving bitcoin transaction propagation by leveraging unreachable nodes. arXiv:2010.15070 [cs.NI]
17. Bitcoin Wiki. Bitcoin core 0.11 (ch 4): P2p network. Last Accessed: 01/04/2021. https://en.bitcoin.it/wiki/Bitcoin_Core_0.11_(ch_4):_P2P_Network
18. Fanti G, Venkatakrishnan SB, Bakshi S, Denby B, Bhargava S, Miller A, Viswanath P (2018) Dandelion++: Lightweight cryptocurrency networking with formal anonymity guarantees. Proc ACM Meas Anal Comput Syst 2(2). https://doi.org/10.1145/3224424
19. Wang L, Pustogarov I (2017) Towards better understanding of bitcoin unreachable peers. CoRR, abs/1709.06837
20. Kaminsky D (2011) Black ops of tcp/ip. Black Hat USA, 44
21. Biryukov A, Tikhomirov S (2019) Deanonymization and linkability of cryptocurrency transactions based on network analysis. In 2019 IEEE European Symposium on Security and Privacy (EuroSP) pp 172–184. https://doi.org/10.1109/EuroSP.2019.00022
22. Mastan ID, Paul S (2018) A new approach to deanonymization of unreachable bitcoin nodes. In: Capkun S, Chow SSM (eds) Cryptology and Network Security. pp. Springer International Publishing, Cham, pp 277–298. ISBN: 978-3-030-02641-7

Federico Franzoni is a researcher in the WiSeCom group at Universitat Pompeu Fabra. He did his PhD in the Department of Information Technologies and Communications at Pompeu Fabra University, Barcelona, Spain. He received his B.Sc. degree in Information Technology and his M.Sc. degree in Computer Science from La Sapienza University in Rome, Italy. His research interests include security, operating systems, networking and blockchain technologies.

Vanesa Daza is an Associate Professor at Pompeu Fabra University, Barcelona, Spain since 2012. She received a Bachelor's Degree in Mathematics from Universitat de Barcelona, and a Ph.D. degree in Mathematics from Universitat Politécnica de Catalunya. She have both worked as a researcher in the industry (Scytl, Spain) as well as academia (Rovira i Virgili University, Spain). She have co-authored more than 30 papers, including international journals and top conferences of cryptography and cybersecurity. Her research interests deal with the use of distributed cryptographic techniques to enhance security and privacy to secure emerging technologies, with special emphasis on blockchain technology. She is an Associate Editor of IEEE Transactions on Information Forensics and Security and IEEE Transactions on Dependable and Secure Computing. Among other positions serving UPF, she chaired the Information and Communication Technologies Department at Pompeu Fabra University.