PERIMETER: A network-layer attack on the anonymity of cryptocurrencies

Maria Apostolaki Cedric Maire Laurent Vanbever

ETH Zürich

Abstract. Cryptocurrencies are widely used today for anonymous transactions. Such currencies rely on a peer-to-peer network where users can broadcast transactions containing their pseudonyms and ask for approval. Previous research has shown that application-level eavesdroppers, namely nodes connected to a large portion of the Bitcoin peer-to-peer network are able to deanonymize multiple users by tracing back the source of transactions. Yet, such attacks are highly visible as the attacker needs to maintain thousands of outbound connections. Moreover, they can be mitigated by purely application-layer countermeasures.

This paper presents a stealthier and harder-to-mitigate attack exploiting the interactions between the networking and application layers. Particularly, the adversary combines her access over Internet infrastructure with application-layer information to deanonymize transactions. We show that PERIMETER is practical in today's Internet, achieves high accuracy in Bitcoin, and generalizes to encrypted cryptocurrencies.

1 Introduction

Anonymity is among the essential properties of any cryptocurrency [38]. The most successful cryptocurrencies today *i.e.,* Bitcoin and Ethereum, are pseudo-anonymous [27]: clients are able to securely transact while using pseudonyms that cannot be trivially mapped to real-world identities. Cryptocurrencies operate using a peer-to-peer (P2P) network of nodes. When a node performs a transaction, it sends the transaction to its peers, which propagate it further. Consequently, an adversary that listens to *all* exchanged messages can map each transaction to the IP address of the node that created it, effectively deanonymizing that node.

Multiple attacks have exploited this transaction broadcasting mechanism to map Bitcoin pseudonyms to their originating IP address [20,22,34,40]. To do so, they use a "supernode": a seemingly regular node that connects to all active Bitcoin nodes and listens to the transaction traffic they relay. [1] Yet, such attacks are highly noticeable [33], as the "supernode" establishes $50 - 117$ new connections to *every* reachable Bitcoin client. Moreover, such attacks can be mitigated by purely application-level countermeasures. For instance, the diffusion broadcast mechanism mitigates the attacks presented in [20,34], while Dandelion [23] and its improvements [28] reduce the effectiveness of the attack presented in [22].

[1] Similar techniques could be applied to Ethereum.

In this work, we introduce PERIMETER: a stealthier harder-to-mitigate network-level attack. PERIMETER relies on an attack vector that has been overlooked: leveraging access to the Internet infrastructure. Connections of any cryptocurrency are inevitably routed over the Internet, thus accessible to multiple Autonomous Systems (ASes) and Internet Exchange Points (IXPs). As a result, a malicious AS or IXP that combines her access to the Internet infrastructure with application-level knowledge can perform a cross-layer deanonymization attack. Our routing analysis on the Bitcoin network reveals that such attacks are practical in today's Internet. Indeed, we found that at least 6 distinct network adversaries can deanonymize more than 35% of the Bitcoin clients (see §5). The PERIMETER attack is stealthier than previous attacks, as it is completely passive (no new connections); and harder to mitigate, as the attacker's power is dependent on the Internet routing protocol (BGP) *i.e.,* not on the application protocol.

PERIMETER is composed of two phases. In the first phase, the attacker eavesdrops on the victim's connections at the *packet-level* to collect information about the transactions the victim propagates to its peers. In the second phase, the attacker analyzes this information to distinguish the victim's transactions.

The attacker eavesdrops on the victim's connections by directly reading each packet's payload *i.e.,* not by establishing connections. In effect, the attack is undetectable and equally effective against NATed nodes, which cannot accept connections. Notably, unlike previous work on network-level attacks [47,17] that require the attacker to control *all* connections of a victim, PERIMETER works with just a fraction. We experimentally show that an adversary intercepting only 25% of the victim's connections can deanonymize it with 70% accuracy (see § 6).

The adversary distinguishes the victim's transactions using anomaly detection (Isolation Forest [35]). The victim's transactions appear as anomalies as they have a distinct propagation pattern. For example, in Bitcoin, the victim will send a transaction that it generated to an unusually high portion of its peers compared to other transactions. Unlike previous work on deanonymizing Bitcoin clients that rely solely on the time difference between announcements of the same transaction across nodes [20,22,33,34], PERIMETER is agnostic to it. As a result, PERIMETER is not sensitive to broadcast protocol changes *e.g.,* diffusion, trickle, etc. Instead, PERIMETER leverages the victim's interactions with its peers to infer whether the victim knew a transaction before its peers.

PERIMETER generalizes to encrypted cryptocurrencies. Taking the popular Ethereum as an example, we observe that an AS or IXP-level adversary is a practical threat for two main reasons. First, similarly to the Bitcoin network, the Ethereum network is affected from the centralization of the Internet traffic. Indeed, we observe that for the majority of clients there are 4 distinct adversaries intercepting 30% of their connections (see §5). Second, a network adversary can infer the victim's peers by eavesdropping on the IP packets, since their header is unencrypted. This combined with the lack of randomness in broadcasting transactions (*e.g.,* diffusion), makes traditional attacks (solved in Bitcoin) such as [34] effective.

To summarize, we make the following key contributions:

- A novel attack vector against anonymity that is effective against Bitcoin (§ 3,§ 4).
- An thorough analysis of the Bitcoin and Ethereum networks from the routing perspective using real-world control-plane data. Our analysis demonstrates the feasibility of such an attack in today's Internet (§5).
- An evaluation of PERIMETER's practicality using both realistic simulations and "in-the-wild" experiments against Bitcoin clients (§6).
- A comprehensive set of deployable countermeasures (§7).

2 Background

In this section, we briefly describe Bitcoin, Ethereum, and BGP.

2.1 Bitcoin workings

Bitcoin is a currency that does not rely on any central authority or trusted party. Instead, Bitcoin relies on a peer-to-peer (P2P) network in which nodes use a consensus mechanism to jointly agree on an append-only log of all the transactions that ever happened, the *blockchain*. Bitcoin users are associated with one or multiple cryptographic pseudonyms, which cannot be trivially mapped to the user's real-world identities. Thus, we say that Bitcoin is pseudonymous. Attempts to map pseudonyms to real-world identities constitute deanonymization attacks.

To transfer funds among each other, Bitcoin clients issue transactions in which they declare the transfer of a certain amount of Bitcoin from their Bitcoin pseudonym to one (or multiple) others. Transactions need to be propagated in the network, verified by all nodes, and eventually added in the blockchain. Upon receiving a new transaction, a Bitcoin client advertises it to its peers using an "inv" message that includes the hash of the transaction. The peers who are unaware of an advertised transaction request it by replying to the advertisement with a "getdata" message that includes the hash of the transaction. Finally, a Bitcoin client sends the transactions to those peers that request it with a "tx" message.

The Bitcoin Core has included two modifications that affect the way transactions are propagated.[2] First, a client advertises transactions with independent, exponential delays to its peers. This broadcast mechanism is called *diffusion* and was introduced as a countermeasure against deanonymization attacks. Second, the diffusion delay that a client adds before an advertisement to a given peer differs depending on who initiated the connection between them. Particularly, a Bitcoin node halves the delay for peers to which it initialized the connections as these are less of a privacy concern [3].

[2] We mention the modifications that are more relevant for our work.

2.2 Ethereum workings

Ethereum supports decentralized applications that are backed by smart contracts: protocols or small pieces of software running on top of the Ethereum network and performing backtrackable and irreversible transactions with no third-party intervention. In the context of Ethereum a transaction is a data structure describing the exchange of Ether signed with the private key corresponding to a users' pseudonym. Similar to Bitcoin, Ethereum relies on a P2P network of nodes and is pseudonymous. In contrast to Bitcoin, though, all Ethereum communications are encrypted [11]. Thus, an on-path eavesdropper cannot read the exchanged messages.

The Ethereum protocol also differs from the Bitcoin's protocol in the way transactions are broadcasted. Ethereum broadcasts newly learned transactions without delay. It also makes use of an advertisement system, but only for a subset of its neighbor peers. In particular, consider a Ethereum (Geth [6]) node with n peers, each time it learns a new transaction, the node broadcasts it to $\lfloor \sqrt{n} \rfloor$ of its peers and then advertises it to the remaining $n - \lfloor \sqrt{n} \rfloor$ peers, excluding those who already know about it.

2.3 BGP workings

The Internet is composed of smaller networks called Autonomous Systems (AS). Each AS contains multiple hosts that are addressed with a unique IP. ASes build physical connections to each other to exchange traffic under certain economic agreements. Oftentimes, ASes also participate in Internet eXchange Points (IXPs). In this case, multiple ASes connect to a single physical location and exchange traffic. BGP[9] is the routing protocol that regulates how IP packets are forwarded in the Internet. Particularity, BGP computes the unidirectional AS-paths along which traffic from each host will reach its destination. ASes and IXPs in this AS-path forward traffic, and thus they can eavesdrop, drop, or delay it.

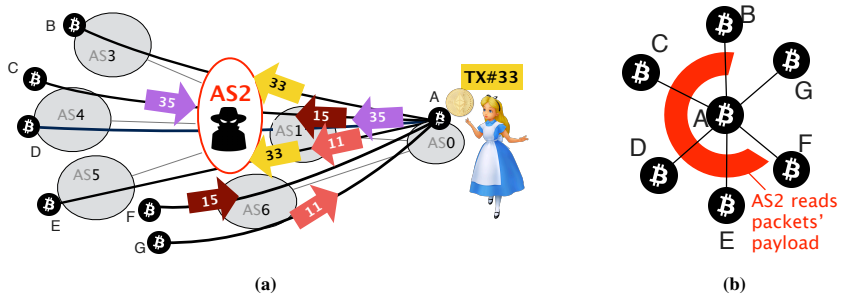
3 Overview

In this section, we give an overview of the PERIMETER attack.We start by briefly describing the attacker's goal, profile, and procedure (§3.1). Next, we illustrate the attack with an example (§3.2) targeting Bitcoin. Finally, we describe how the attack could be generalized to Ethereum (§3.3).

3.1 PERIMETER at a high-level

Attacker's goal The attacker's goal is to deanonymize a specific node, meaning to map the IP of a victim node to the transaction(s) it created.[3] Concretely, the attacker's goal is to compute a set of transactions that contains (most of) the victim's transaction(s) (*i.e.,* maximize true positives) and as few as possible transactions created by other

[3] Such an attack is very harmful to the victim because an attacker can often link all other transactions the victim made to the deanonymized one [39]



(a) (b)

Fig. 1: (a) From the networking viewpoint, the attacker (AS2) naturally *i.e.,* according to BGP, intercepts some of the victim's connections. (b) From the application viewpoint, the attacker (partially) surrounds the victim without establishing any new connection. Surrounding the victim allows the attacker (AS2) to read the unencrypted Bitcoin messages the victim node A sends and receives.

clients (*i.e.,* minimize false positives). We refer to this set of transactions as the victim's *anonymity set*.

Attacker's profile The attacker is an Autonomous System (AS) or Internet eXchange Point (IXP) that naturally (*i.e.,* according to BGP's calculations) intercepts any direction of X% of the victim's connections and knows the victim's IP.[4] Due to the centralization of the Internet traffic, many ASes and IXPs intercept a large portion of a host's connections even if they are not their direct provider, as we show in §5.

Attack procedure The attack consists in eavesdropping the victim's connections and analyzing collected data to distinguish the victim's transaction(s). Concretely, the adversary first leverages her position in the Internet to gain visibility over the transactions that the victim propagates.We refer to this process as *surrounding* since the adversary tries to create a logical circle around the victim across which she can observe the incoming and outgoing information. Next, the adversary computes statistics on the transactions the victim advertises and uses anomaly detection to find the victim's transactions. Useful statistics include the number of times the victim or its peers sent or received a transaction.

3.2 PERIMETER in action

An example scenario Fig. 1a illustrates how an attacker running PERIMETER can deanonymize Bitcoin transactions. This network is composed of seven ASes (AS0 - AS6), some of which host Bitcoin clients (nodes A-G). Traffic between each pair of nodes is forwarded following the AS-path that BGP calculates. As a result, AS2 intercepts the connections between node A and nodes B, C, D and E. Assume that AS2 is malicious and aims at deanonymizing Alice's transactions. AS2 knows the IP of the

[4] Finding the IP of a person is practical as it is revealed every time this person visits a website or an application *e.g.,* skype call.

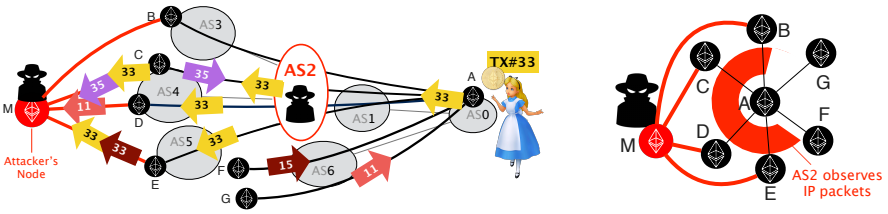


Fig. 2: To deanonymize Alice's transaction in Ethereum, the attacker (AS2) connects to some of the victim's peers. AS2 infers some of the victim's peers' IPs by eavesdropping on the victim's connections. (b) In effect, the attacker indirectly surrounds the victim from the application viewpoint.

node on which Alice runs her Bitcoin wallet, namely the IP of node A. Thus, AS2 aims at mapping node A to the transaction(s) it generates, that is TX#33 in this example.

PERIMETER attack on Bitcoin AS2 eavesdrops on the victim's connections that she naturally intercepts and creates the initial anonymity set from the transactions that node A propagates *i.e.,* TX #15, TX #11, TX #35, and TX #33. From the application viewpoint, AS2 has formed a (partial) logical circle around the victim, as illustrated in Fig. 1b, while being passive. Next, AS2 tries to reduce the size of the anonymity set by removing transactions that are most likely not generated by the victim. AS2 knows that a Bitcoin node only receives transactions it requests, and only requests transactions it does not know already. Thus, AS2 excludes TX #35 from the anonymity set as AS2 has observed node A receiving TX #35 from node C. Observe that AS2 cannot use the same technique for TX #15 and TX #11 as AS2 does not intercept the connection to nodes F and G from which node A received them. Instead, AS2 uses anomaly detection to find the victim's transaction, as it appears as an anomaly with respect to its propagation pattern. For instance, the number of peers that requested TX #33 from node A is higher for TX #33 than for any other transaction. We elaborate on the anomaly detection procedure and the features used in §4.

3.3 Generalizing PERIMETER to Ethereum

Fig. 2a illustrates the same attack scenario as before but with nodes A-G belonging to the Ethereum network. AS2 could use PERIMETER to deanonymize node A.

Unlike Bitcoin, Ethereum connections are encrypted, meaning that AS2 cannot directly read the content of the messages the victim exchanges with its peers. AS2 uses the observation made in [34] to deanonymize Bitcoin clients. Particularly, according to [34], a node can be uniquely identified in a single session by their directly connected neighboring nodes. Unlike [34] that uses a Bitcoin-specific flaw to infer connections, AS2 can infer the IP addresses of the victim's peers by reading the unencrypted headers of the packets the victim node A inevitably sends and receives.[5] After connecting to

[5] Ethereum facilitates connecting to a client using its IP (*i.e.,* discovery v4 UDP packet).

some of the victim's peers, distinguishing the victim's transactions (across those its peers propagate) is strictly more straightforward than for Bitcoin. That is the case as most Ethereum nodes (geth version [6]) advertise the new transactions immediately to their peers. From the application viewpoint, the adversary has again partially surrounded the victim, as seen in Fig. 2b.

4 PERIMETER workings

Having described the PERIMETER attack at the high-level in §3, we now elaborate on PERIMETER's technical details. Concretely, we describe how the attacker *(i)* distinguishes Bitcoin traffic (§4.1); *(ii)* retrieves propagated transactions (§4.2); and *(iii)* uses anomaly detection (Isolation Forest) to find the victim's transaction(s) (§4.3). Finally, we discuss the features the attacker uses (§4.4).

4.1 Recognizing Bitcoin traffic

The adversary surrounds the victim node and reads the data exchanged in the Bitcoin connections to create the initial anonymity set. To do so, the adversary first needs to distinguish Bitcoin traffic across all the connections she intercepts. The adversary can easily distinguish Bitcoin traffic since most clients use a particular TCP port, *i.e.,* 8333. Notably, the adversary can recognize the Bitcoin connections, even between clients using another TCP port. To do so, the adversary searches on the packet payload to find known Bitcoin message types, *e.g.,* "inv" or "getdata". Indeed, string searching practical in commodity hardware [32]. Observe that the adversary performs string search at most in a single packet per connection. Once the adversary finds a Bitcoin message in a packet of a connection, she can use a filter that matches on the 4-tuple of the TCP connection (*i.e.,* IP addresses and TCP ports) to distinguish it.

4.2 Creating the initial anonymity set

To create the initial anonymity set, the attacker needs to distinguish all transactions that the victim itself or its peers have advertised. This is challenging as Bitcoin messages can be split among multiple packets and those packets can be re-ordered, lost, and re-transmitted while being transferred in the Internet. In effect, concatenating each Bitcoin connection's payloads (packet stream) would not result in the complete list of messages that the corresponding clients exchanged (message stream). To reconstruct the message stream, the adversary can use tools such as GoPacket [7] that leverage the sequence number contained in the TCP header.

Next, the adversary includes in the anonymity set the hashes of the transactions that are included in three types of messages, namely "inv", "getdata" and "tx". Finally, the adversary calculates statistics per transaction hash. Particularity, she calculates the number of "inv","getdata" and "tx" that are sent and received per transaction.

4.3 Analyzing data

Having collected the initial anonymity set, the adversary needs to reduce it to the transactions that the victim created. Doing so is challenging for two reasons. First, the number of transactions the victim propagates is orders of magnitudes higher than those that it created. Second, the adversary does not have ground truth to train on. To address these challenges, the adversary formulates the problem to an unsupervised anomaly detection problem, meaning a problem that requires identifying data points that differ from the norm (i.e., anomalies) in an unlabeled dataset. The victim's transactions appear as anomalies in the anonymity set because they are a tiny minority (compared to all transactions the victim propagates), and they exhibit different propagation patterns, *e.g.,* the victim will propagate the transaction it generates to more peers compared to other transactions.

The attacker uses an Isolation Forest (IF) ([35],[36]) to solve this unsupervised anomaly detection problem, since IF is more efficient and interpretable than clustering-based approaches or neural networks. Concretely, IF is significantly more computationally efficient than distance-based methods, including classical nearest-neighbor and clustering-based approaches. This is because IF is a tree-based machine learning algorithm that directly identifies anomalies by isolating outliers in the data rather than first defining the normal behavior and calculating point-based distances. In effect, IF can scale well with large datasets, making it suitable for real-time online applications. Finally, in contrast to neural network methods, such as autoencoders, IF is easy to interpret, and it is not too sensitive to parameter tuning.

IF achieves this by building an ensemble of decision trees to partition the data points. To create these trees, IF recursively generates partitions by randomly selecting a feature and then selecting a random split value between the minimum and the maximum value of the selected feature. The number of required random splits to isolate a sample averaged over a forest of such random trees determines the normality of a sample. IF leverages the observation that anomalies are more natural to isolate, and thus they need fewer splits on average than normal data points.

4.4 Feature selection

We started our feature investigation with a pool of features, including some timing-related and some interaction-related (*i.e.,* related to the interaction between the victim and its peers). Using cross-validation in our simulation runs (see § 6.1), we selected 3 interaction-related features: *(i)* number of "getdata" messages; *(ii)* number of "tx" messages; and *(iii)* the portion of clients which requested a transaction. Next, we describe the features in detail and explain why they allow the victim's transaction(s) to stand out as anomalies.

The number of "getdata" messages that the victim received per transaction: This is equivalent to the number of times the victim sent a transaction. Thus, the adversary can capture this feature independently of the direction of the victim's connections she intercepts. A Bitcoin client will only send a "getdata" for an advertised transaction if it has not received this transaction before. Thus, the victim is expected to receive more "getdata" for a transaction it created, as its peers are unlikely to have received it by others.

The number of "tx" messages the victim received per transaction: If the victim received a transaction from one of its peers, then the victim could not have created it. That is because, in order for the victim to receive a transaction, it should have requested this transaction from its peer, and thus, it should not have known this transaction beforehand. The number of "tx" the victim received for a transaction is equivalent to the number of "getdata" the victim sent. In effect, an AS-level (or IXP-level) adversary would be able to calculate this feature independently of the direction of traffic she intercepts, namely *to* or *from* the victim node.

The portion of clients that requested a transaction from the victim across those the victim advertised this transaction to: This feature is similar to the number of "getdata" with one critical difference. It considers that because of diffusion, the victim might delay advertising its transaction to some of the peers so much that they learn it from elsewhere. The victim's transaction will have a high request/advertisement ratio because the victim knows about the transaction much earlier than its peers.

5 PERIMETER's practicality

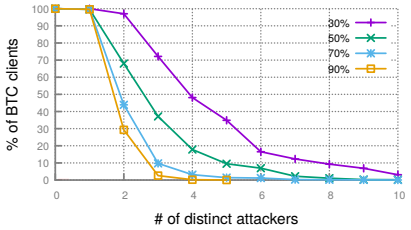
As we described in §3, an effective PERIMETER attacker needs to naturally intercept some of the victim's connections. In this section, we show that this attacker model is practical, taking into consideration both real-world Internet routing and the two biggest cryptocurrency peer-to-peer networks, namely the Bitcoin and the Ethereum network. To that end, we first investigate how likely it is for a given client to be vulnerable to PERIMETER. We found that for 50% of the Bitcoin (60% of the Ethereum) clients, there are at least 4 distinct network adversaries that can intercept 30% of their connection. Second, we investigate how likely it is for a random transaction to be deanonymized. We found that only 5 network adversaries (if they were conspiring) would be able to deanonymize the majority of transactions created in Bitcoin. We describe our methodology in §5.1 before we summarize our results in §5.2.

5.1 Methodology

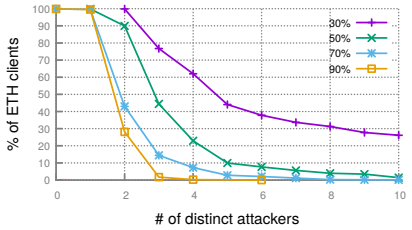
To realistically evaluate the practicality of the PERIMETER attack, we simulated BGP[9], the default routing protocol in the Internet. Particularly, for each pair of ASes in the Internet, we compute the BGP AS-path: the sequences of ASes and IXPs that can intercept packets sent by clients hosted in this AS pair. We then calculated the ability of various ASes and IXPs to perform various-powered PERIMETER attacks against the Bitcoin and Ethereum clients. Notably, we augment the routing analysis of the Bitcoin network presented in [17] by adding IXP links and by analyzing the Ethereum network.

We used three datasets for our evaluation: *(i)* the IPs of the Ethereum and Bitcoin clients; *(ii)* the BGP advertised routes; and *(iii)* the publicly-available economic relationship among ASes and IXPs.

Ethereum & Bitcoin IPs to ASes We fetched the IPs of the Bitcoin and Ethereum from publicly available data [12] [13]. We removed onion addresses as we could not assign them to actual IPs. Next, we inferred the most-specific prefix and the AS hosting each



(a) For 35% of the Bitcoin clients there are at least 5 distinct attackers that can intercept 30% of their connections.



(b) For 37% of the Ethereum clients there are at least 6 distinct attackers that can intercept 30% of their connections.

Fig. 3: Both Bitcoin and Ethereum are vulnerable to PERIMETER's attacker model.

Bitcoin and Ethereum client. To that end, we processed almost a million BGP routes (covering all Internet prefixes) advertised on BGP sessions maintained by 6 RIPE BGP collectors [10] (rrc00- rrc05). We do the mapping by associating each prefix to the origin AS advertising it.

AS-level topology and forwarding paths To infer an AS-level topology, we used the economic relationships between ASes provided by CAIDA [25]. An AS-level topology is a directed graph in which each node corresponds to an AS, and each link represents an inter-domain connection between two neighboring ASes. Each link is also labeled with the business relationship between the two ASes (customer, peer, or provider). We augmented our AS-level topology with IXP links provided by CAIDA [26] following the methodology in [15,37].

Our augmented AS-level topology is composed of $\sim 67K$ ASes, more than ~ 700 IXPs, and $\sim 4M$ links. Our datasets were collected in September 2019. We computed the actual forwarding paths on our AS-level topology following the routing tree algorithm described in [30].

5.2 Findings

Using the Internet topology described in §5.1, we calculated how vulnerable individual clients are to PERIMETER and how likely it is for a transaction to be deanonymized.

The majority of Bitcoin clients are vulnerable to PERIMETER by multiple potential attackers. We calculated the number of distinct attackers able to intercept a fraction of the potential victims' connections. We summarize our results in Fig. 3a. The x-axis corresponds to the number of distinct attackers that can perform a PERIMETER attack against the portion of the Bitcoin clients shown in the y-axis. We consider four attack types depending on the fraction of the victim's traffic that the attacker intercepts. Specifically, we consider attackers intercepting 30%, 50%, 70%, and 90%, which correspond to different lines in the plots. As expected, all Bitcoin clients are vulnerable to PERIMETER by their own provider, which observes $> 90\%$ of their connections. Interestingly though, $> 90\%$ of all Bitcoin clients are also vulnerable to PERIMETER by at least one more network adversary. Moreover, we observe that for 50% of the Bitcoin clients, there are

at least 4 attackers able to intercept 30% of their connections. This is worrying as such adversaries can deanonymize Bitcoin clients with at least 70% accuracy, as we observe from our experiments in the Bitcoin Mainnet (see §6). Worse yet, for 20% of the Bitcoin clients, there are at least 4 potential attackers that can perform the PERIMETER attack leveraging their access to 50% of the victims' connections.

PERIMETER's attacker model is practical in the Ethereum network. We plot the same results for Ethereum in Fig. 3b. We observe that Ethereum is slightly more vulnerable than Bitcoin to a passive AS-level (or IXP-level) adversary. Particularly, we observe that for most clients, there are 4 distinct network adversaries intercepting 30% of their connections. Observe that these adversaries can almost effortlessly infer 30% of the clients' peers. This is worrying considering that geth [6], the most used Ethereum version [12], does not implement diffusion or any other randomized broadcast mechanism.

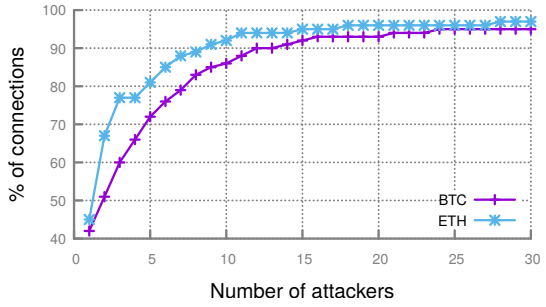


Fig. 4: 5 network adversaries intercept 72% (80%) of all possible Bitcoin (Ethereum) connections.

Few well-established attackers can perform a network-wide deanonymization attack. Fig. 4 illustrates the cumulative distribution of the percentage of the connections as a function of the number of potential network adversaries intercepting it. We observe that only 10 ASes together intercept 90% of the Ethereum clients and 85% of the Bitcoin clients. If those 10 ASes and IXPs decided to collude, they would be able to deanonymize 85% of all transactions in Bitcoin and able to infer at least 90% of the Ethereum peer-to-peer graph. This is especially alarming considering that the attack is entirely passive; thus, there is no reputation risk involved in performing it.

6 PERIMETER's effectiveness

We evaluate the effectiveness of PERIMETER in simulation (§6.1) and in the wild (§6.2). We found that an attacker intercepting 25% of the victim's connections can deanonymize a client with 70% accuracy in the Bitcoin Mainnet. Unsurprisingly, the PERIMETER attack appears even more effective in simulation.

6.1 PERIMETER in simulation

We evaluated PERIMETER using a realistic simulation whose delays we have tuned based on Internet-wide measurements. We describe our methodology before we describe our results.

Simulator We modeled the entire Bitcoin networks by extending the realistic event-driven simulator used in [17]. We used the 0.19.1 version of the Bitcoin Core as a reference for the behavior of the Bitcoin clients. Among other implementation details, we simulated the Poisson delay that a node waits before advertising a new transaction (diffusion) and its preference to outgoing connections in advertising and in requesting transactions [3]. We simulated all nodes whose IPs were reachable, and we could locate in the Internet as described in § 5.

Simulating Internet delays To realistically model Internet delays among clients in our simulation, we leveraged the RIPE Atlas platform. RIPE Atlas [1] is a data collection system composed of a global network of devices, called probes that can actively perform Internet measurements. In particular, to estimate the delay between each pair of Bitcoin nodes, we measure the delay between probes in the ASes hosting these Bitcoin nodes. Indeed, Internet delay between two particular hosts located in any AS-pair is representative of the delay between any pair of hosts in the same AS-pair. That is because the Internet path between any pair of hosts in the same AS-pair is common. We performed ping measurements for each pair of ASes say (ASA, ASB) in which there are at least two Bitcoin clients (*e.g.,* one Bitcoin client in ASA and one in ASB) and at least one RIPE probe available (*i.e.,* either in ASA or in ASB). If multiple probes existed in the same AS, we used one for each prefix in which at least one client is hosted. We perform each measurement at least three times and use the median delay. Our measurement campaign lasted 7 hours and included $\sim 50K$ pings. We archived delay measurements available from RIPE atlas [2] to add the delays of AS-pairs, which we could not measure ourselves. Together these delay measurements cover 72% of the Bitcoin connections.

We configure the delay of each node pair in the simulation with a randomly-selected value across the delays measured in the corresponding AS-pairs. We validated our augmented simulator by ensuring that the median transaction propagation delay aligns with the value reported in [14].

Procedure We simulated a total of 10000 transactions, among which 100 were created by the victim client. We use 70% of all transactions for training and 30% for testing. For feature selection, we use 5-fold cross-validation on the training set. We run the attack assuming the adversary intercepts a fraction of the victim's connection *i.e.,* 25%, 50%, 75%, 100%.

Results We summarize our results in Table 1. We observe that an attacker can always (*i.e.,* almost independently of the percentage of connections she intercepts) deanonymize the victim with 100% true positive and low false-positive rate. This is expected because the simulated environment is idealized. In particular, *all* clients run the same code, are benign and are in the same condition concerning load. Such an environment creates a straightforward case for anomaly detection.

Simulation	25%	50%	75%	100%
true positives	1	1	1	1
false positives	0.002%	0.002%	0.001%	0%

Table 1: In simulation an adversary deananonymizes the victim with 100% accuracy even when intercepting 25% of its connections.

Mainnet	25%	50%	75%	100%
true positives	0.7	0.9	0.9	1
false positives	0.002%	0.003%	0.003%	0.0%

Table 2: In the wild an adversary deanonymizes the victim with 90% accuracy when intercepting 50% of its connections.

6.2 PERIMETER in the wild

We evaluated PERIMETER on the actual Bitcoin network. We describe our methodology before we describe our results.

Methodology For our in-the-wild experiment, we used as victim a Bitcoin node version 0.19.1 of the Bitcoin Core running. Since we only attack our own node, our experiment is ethical: we did not disturb the normal operation of the Bitcoin network in any way. We configured our victim to not listen for incoming connections but instead only connect to a predefined set of peers randomly selected across those in [13].[6] We capture a total of $\sim 30K$ of transactions, among which 10 transactions were created by our victim. As the attack is completely passive, we use the same transactions to measure the effectiveness of various powered adversaries. In particular, we run the attack assuming the adversary intercepts a fraction of the victim's connection *i.e.,* 25%, 50%, 75%, 100%.

We split the resulting dataset into the training and testing sets. We used all the victim's transactions and 30% of the transactions from other clients as the testing set. We included all of the victim's transactions in the testing to have a more accurate estimate of true positives. In any case, the victim's transactions are too few to affect the training of the model. We used the remaining 70% of transactions from other clients as the training set. Finally, we use the features we selected in the simulation, and we describe in §4.4.

Results Table 2 summarizes our results. We observe that an adversary intercepting only 25% of the victim's connections can deanonymize it with 70% true positives, and only 0.002% false positives. Moreover, an adversary intercepting 50% of a client's connections (or more) can deanonymize the victim with 90% accuracy (or above). As a baseline, consider that previous attacks using "supernodes" report accuracies of 11%-60% [20] and 75% [22] thus lower than PERIMETER. This demonstrates the effectiveness of a network-layer attacker exploiting her access over the Internet infrastructure.

[6] We do not allow incoming connections to prevent attacks from light clients during the experiment.

7 Countermeasures

The PERIMETER attack poses a serious and practical threat to the anonymity properties of Bitcoin, as shown in §5 and §6. We argue that such a threat should and can be avoided for current and future cryptocurrencies by employing the following countermeasures.

Encrypting traffic One of the most critical enablers of the PERIMETER attack is that traffic is routed over the Internet unencrypted. Undoubtedly, encrypting Bitcoin's traffic would make the currency more anonymous, at least from the PERIMETER's perspective. Still, encryption alone cannot adequately mitigate the threat of passive AS-level adversaries. Observe that PERIMETER attack generalizes to Ethereum whose traffic is encrypted, even though in this case, the attacker also needs to establish new connections.

Using fake peers PERIMETER generalizes to encrypted cryptocurrencies (*e.g.,* Ethereum) because of the networking footprint of its clients, that is, the IPs of the clients' peers (fake peers). Particularly, a networking attacker can infer a client's peers by eavesdropping on this client's connections. Inferring some of a client's peers is critical for its deanonymization [34]. To prevent such an attack, a client could establish connections to peers with which it does not interact in practice, effectively persuading a potential attacker to connect to irrelevant clients. To achieve this, the client should not request or store any transaction from fake peers; neither should it not advertise new transactions to them. In effect, the client obfuscates its footprint from a networking attacker.

Obfuscating the client's state One of the key features used to deanonymize Bitcoin clients in the PERIMETER attack is whether the victim requested (or received) a particular transaction from any of its observed peers. By requesting a transaction, the client reveals to a networking attacker (or potentially malicious client) that they do not know about a transaction and thus that they have not created it. As a result, the adversary can safely exclude some transactions, effectively decreasing the initial anonymity set. This is also true for Ethereum geth [6]. To avoid this, a client should request also the transactions it creates from peers who advertise them. While by doing so the client increases its load without learning anything new, it also deprives potential attackers of an extremely effective feature. Notably, obfuscating the transactions a client knows by requesting them is aligned with obfuscating the transactions a light client is interested in by requesting more transactions (*i.e.,* using Bloom Filters in Bitcoin's BIP37 [31]).

Routing-aware transactions' requests Instead of requesting more transactions to obfuscate its state, a client can achieve a similar effect by carefully selecting the peer from which it requests an unknown transaction. Particularly, a client should request transactions in a routing aware manner, meaning avoid requesting multiple transactions from clients whose connections are intercepted by the same AS or IXP. In effect, an adversary is unlikely to have an accurate view of which transactions the victim knew.

Routing-aware transactions' advertisement While PERIMETER does not directly use timing information, previous works [34,22,20] have shown the importance of obfuscating the first-ever propagation of a transaction. In fact, this need has motivated countermeasures, such as the adoption of diffusion in the Bitcoin Core and the creation of Dandelion[28]. We argue that such improvements need to also account for AS-level adversaries. Particularly in diffusion, one could increase the delay for clients whose path contains a very common AS or IXP. Similarly, the first node to advertise a transaction in

the Dandelion protocol could be selected (in addition to the current criteria) such that the created traffic does not often traverse the same AS or IXP.

Using Tor or VPN services The goal of the PERIMETER attacker is to link transactions to IP addresses. Thus, if a client manages to obfuscate its IP address by using Tor or a VPN service, it should be protected against some of the potential attackers. Unfortunately, this statement is only partially true. Regarding Tor, a network adversary can easily prevent the client from using Tor either by exploiting the Dos mechanism [21] or by merely dropping the corresponding traffic. Observe that the latter is possible as the IPs of all Tor relays are publicly known, and the adversary might intercept the corresponding connections. Even if the client manages to use Tor, it would still be vulnerable to deanonymization by a network attacker that leverage timing analysis [46]. On the contrary, using a VPN service would successfully obfuscate one's IP, but the attack would reveal the connection between the VPN IP and the transactions.

8 Related Work

Deanonymizing cryptocurrency transactions Researchers have studied Bitcoin's anonymity properties from two angles: the blockchain analysis and the traffic analysis of the P2P network. From the blockchain-analysis angle: several papers have shown that linking transactions made by the same user relying on publicly available blockchain data is possible [16,41,42,43,39] even across sessions. These works are orthogonal to ours. From the traffic-analysis angle: several papers have shown that linking transactions to IPs is possible by analyzing data collected from one or multiple "supernodes" [20,22,34,40], which establish connections to all reachable clients. Unlike, PERIMETER such attacks are visible as the attacker needs to maintain thousands of outbound connections. Moreover, these attacks can be mitigated by existing techniques such as diffusion, increased delay to inbound connections [3] and Dandelion [23,28].

AS-level adversaries AS-level attacks can be active or passive. In an active attack, the adversary performs BGP hijacks. Active adversaries can partition the Bitcoin network [17], deanonymize Tor [46], and compromise certificate authorities' infrastructure [19]. While effective, active AS-level adversaries are highly visible. In a passive attack, the adversary operates on traffic that she naturally intercepts. Passive adversaries can eclipse clients [47] and delay blocks [17]. PERIMETER is orthogonal to the above works as it acts against anonymity while being completely invisible (passive attacker).

Measuring cryptocurrency networks Apostolaki et al. [17] presented the first analysis of the Bitcoin network from the routing perspective. Our analysis augments this by including IXPs in the AS-level topology and by analyzing the Ethereum network. Gencer et al. [29] analyzed both Ethereum and Bitcoin from the bandwidth, availability and geographic distribution perspective but not from the routing perspective. Finally, Saad et al. [45] presented an measurement analysis on the AS distribution, location, and performance of Bitcoin clients, again not considering Internet routing.

Countermeasures To the best of our knowledge, none of the existing countermeasures against known attacks protects against PERIMETER. Countermeasures against deanonymization attacks such as Dandelion [23,28] do not prevent AS-level attacks as the selection of the first peer who receives a new transaction is independent of the

AS-level topology. Relay networks such as Falcon [4], SABRE [18], and FIBRE [5] are irrelevant to PERIMETER as they focus on block propagation. Mixing protocols [24,44] allow users to obscure transaction history, but cannot prevent a PERIMETER attacker from mapping the IP of a node with a transaction that it created. Finally, a recent modification in Bitocoin Core [8] reduces the chances of a client to select peers from the same AS by improving IP bucketing. While this is effective against [47], it cannot prevent PERIMETER. That is because this selection only affects outgoing connections, and most importantly it does not consider the AS-path.

9 Conclusion

This paper presented PERIMETER, the first passive AS-level deanonymization attack that is practical and effective against Bitcoin. We showed that PERIMETER is stealthier than previous deanonymization attacks while achieving higher accuracy. Based on real-world data, we showed that Bitcoin and Ethereum are very vulnerable to PERIMETER's attacker model. While PERIMETER poses a severe threat to Bitcoin and similar cryptocurrencies, we also explained a comprehensive list of deployable countermeasures.

References

1. About: What is RIPE Atlas? https://atlas.ripe.net/landing/about/
2. Announcing Daily RIPE Atlas data archives. https://labs.ripe.net/Members/petros_gigis/announcing-daily-ripe-atlas-data-archives
3. Bitcoin Core diffusion delay, https://github.com/bitcoin/bitcoin/blob/da4cbb7927497ca3261c1504c3b85dd3f5800673/src/net_processing.cpp#L3813
4. Fast Internet Bitcoin Relay Engine, https://www.falcon-net.org
5. FIBRE, https://bitcoinfibre.org/
6. Go Ethereum: Official Go implementation of the Ethereum protocol. https://github.com/ethereum/go-ethereum
7. GoPacket. https://github.com/google/gopacket
8. p2p: supplying and using asmap to improve IP bucketing in addrman, https://github.com/bitcoin/bitcoin/pull/16702
9. RFC 1267 – Border Gateway Protocol 3 (BGP-3). https://tools.ietf.org/html/rfc1267
10. RIPE RIS Raw Data. https://www.ripe.net/data-tools/stats/ris/ris-raw-data
11. The RLPx Transport Protocol. https://github.com/ethereum/devp2p/blob/master/rlpx.md
12. Ethereum Mainnet Statistics. https://www.ethernodes.org (2020)
13. GLOBAL BITCOIN NODES DISTRIBUTION. https://bitnodes.io/ (2020)
14. Propagation of Transactions and Blocks. https://dsn.tm.kit.edu/bitcoin/#propagation (2020)
15. Ager, B., Chatzis, N., Feldmann, A., Sarrar, N., Uhlig, S., Willinger, W.: Anatomy of a large european ixp. In: Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication. p. 163–174. SIGCOMM '12, ACM (2012). https://doi.org/10.1145/2342356.2342393, https://doi.org/10.1145/2342356.2342393
16. Androulaki, E., Karame, G.O., Roeschlin, M., Scherer, T., Capkun, S.: Evaluating user privacy in bitcoin. In: International Conference on Financial Cryptography and Data Security. pp. 34–51. Springer (2013)
17. Apostolaki, M., Zohar, A., Vanbever, L.: Hijacking bitcoin: Routing attacks on cryptocurrencies. In: S&P '17 (May). https://doi.org/10.1109/SP.2017.29
18. Apostolaki, M., Marti, G., Müller, J., Vanbever, L.: Sabre: Protecting bitcoin against routing attacks. In: Proceedings of the 26th Annual Network and Distributed System Security Symposium. p. 02A1. Internet Society (2019)
19. Birge-Lee, H., Sun, Y., Edmundson, A., Rexford, J., Mittal, P.: Bamboozling certificate authorities with {BGP}. In: 27th {USENIX} Security Symposium ({USENIX} Security 18). pp. 833–849 (2018)
20. Biryukov, A., Khovratovich, D., Pustogarov, I.: Deanonymisation of clients in bitcoin p2p network. In: CCS '14
21. Biryukov, A., Pustogarov, I.: Bitcoin over tor isn't a good idea. In: 2015 IEEE Symposium on Security and Privacy. pp. 122–134. IEEE (2015)
22. Biryukov, A., Tikhomirov, S.: Deanonymization and linkability of cryptocurrency transactions based on network analysis. In: EuroS&P '19
23. Bojja Venkatakrishnan, S., Fanti, G., Viswanath, P.: Dandelion: Redesigning the bitcoin network for anonymity. POMACS (2017)

24. Bonneau, J., Narayanan, A., Miller, A., Clark, J., Kroll, J.A., Felten, E.W.: Mixcoin: Anonymity for bitcoin with accountable mixes. In: International Conference on Financial Cryptography and Data Security. pp. 486–504. Springer (2014)
25. The CAIDA AS relationship dataset - 20191001. http://data.caida.org/datasets/as-relationships/serial-1/
26. The caida ixps dataset - 201910. http://data.caida.org/datasets/ixps/ix-asns_201910.jsonl, accessed: 2020-03-12
27. Extance, A.: The future of cryptocurrencies: Bitcoin and beyond. Nature News **526**(7571), 21 (2015)
28. Fanti, G., Venkatakrishnan, S.B., Bakshi, S., Denby, B., Bhargava, S., Miller, A., Viswanath, P.: Dandelion++: Lightweight cryptocurrency networking with formal anonymity guarantees. POMACS (2018)
29. Gencer, A.E., Basu, S., Eyal, I., Van Renesse, R., Sirer, E.G.: Decentralization in bitcoin and ethereum networks. In: International Conference on Financial Cryptography and Data Security. pp. 439–457. Springer (2018)
30. Goldberg, S., Schapira, M., Hummon, P., Rexford, J.: How secure are secure interdomain routing protocols. ACM SIGCOMM Computer Communication Review **40**(4), 87–98 (2010)
31. Hearn, M., Corallo, M.: Connection bloom filtering. bitcoin improvement proposal 37 (2012)
32. Jepsen, T., Alvarez, D., Foster, N., Kim, C., Lee, J., Moshref, M., Soulé, R.: Fast string searching on pisa. In: Proceedings of the 2019 ACM Symposium on SDN Research. pp. 21–28 (2019)
33. Khalilov, M.C.K., Levi, A.: A survey on anonymity and privacy in bitcoin-like digital cash systems. IEEE Communications Surveys & Tutorials **20**(3), 2543–2585 (2018)
34. Koshy, P., Koshy, D., McDaniel, P.: An analysis of anonymity in bitcoin using p2p network traffic. In: International Conference on Financial Cryptography and Data Security. pp. 469–485. Springer (2014)
35. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation forest. In: 2008 Eighth IEEE International Conference on Data Mining. pp. 413–422. IEEE (2008)
36. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation-based anomaly detection. ACM Transactions on Knowledge Discovery from Data (TKDD) **6**(1), 1–39 (2012)
37. Luckie, M., Huffaker, B., Dhamdhere, A., Giotsas, V., Claffy, K.: As relationships, customer cones, and validation. In: Proceedings of the 2013 conference on Internet measurement conference. pp. 243–256 (2013)
38. Matetic, S., Wüst, K., Schneider, M., Kostiainen, K., Karame, G., Capkun, S.: BITE: Bitcoin lightweight client privacy using trusted execution
39. Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G.M., Savage, S.: A fistful of bitcoins: characterizing payments among men with no names. In: Proceedings of the 2013 conference on Internet measurement conference. pp. 127–140 (2013)
40. Neudecker, T., Hartenstein, H.: Could network information facilitate address clustering in bitcoin? In: International conference on financial cryptography and data security. pp. 155–169. Springer (2017)
41. Ober, M., Katzenbeisser, S., Hamacher, K.: Structure and anonymity of the bitcoin transaction graph. Future internet **5**(2), 237–250 (2013)
42. Reid, F., Harrigan, M.: An analysis of anonymity in the bitcoin system. In: Security and privacy in social networks, pp. 197–223. Springer (2013)
43. Ron, D., Shamir, A.: Quantitative analysis of the full bitcoin transaction graph. In: International Conference on Financial Cryptography and Data Security. pp. 6–24. Springer (2013)
44. Ruffing, T., Moreno-Sanchez, P., Kate, A.: Coinshuffle: Practical decentralized coin mixing for bitcoin. In: European Symposium on Research in Computer Security. pp. 345–364. Springer (2014)

45. Saad, M., Cook, V., Nguyen, L., Thai, M.T., Mohaisen, A.: Partitioning attacks on bitcoin: Colliding space, time and logic. Tech. rep., Tech. Rep (2019)
46. Sun, Y., Edmundson, A., Vanbever, L., Li, O., Rexford, J., Chiang, M., Mittal, P.: {RAPTOR}: Routing attacks on privacy in tor. In: 24th {USENIX} Security Symposium ({USENIX} Security 15). pp. 271–286 (2015)
47. Tran, M., Choi, I., Moon, G.J., Vu, A.V., Kang, M.S.: A stealthier partitioning attack against bitcoin peer-to-peer network. In: S&P '20