Deanonymisation in Ethereum Using Existing Methods for Bitcoin

February 7, 2018

Robin Klusman
Security and Network Engineering
University of Amsterdam
robin.klusman@os3.nl

Tim Dijkhuizen
Security and Network Engineering
University of Amsterdam
tim.dijkhuizen@os3.nl

***Abstract*—This paper aims to apply two existing deanonymisation techniques developed for the Bitcoin network to Ethereum. Ethereum is a blockchain based platform like Bitcoin, but there are some significant differences with regard to their peer-to-peer network and blockchain structure. The first attack evaluated in this paper is an attempt to reveal IP addresses of Bitcoin users by monitoring the peer-to-peer network. The second attack aims to cluster groups of addresses belonging to the same user by analysing the Bitcoin blockchain. It proves difficult to apply these methods to Ethereum due to the differences between both networks in the volatility of entry nodes and the way transactions are handled. However, by exploiting some of the specifics of the Ethereum network, such as the peer selection protocol or the bootnodes, similar attacks are potentially feasible.**

I. Introduction

Blockchain technology has gained significant popularity over the recent years since the launch of Bitcoin in 2009 based on the paper by Nakamoto from the year prior [1]. With the massive popularity of Bitcoin, other blockchain based platforms started to emerge, one of which is Ethereum that was launched in 2015 [2]. Ethereum is a blockchain-powered application platform that is able to run smart contracts which consist of any code made for the Ethereum Virtual Machine (EVM). Ethereum has its own digital currency called Ether (ETH), which is also used to buy the *gas* needed for the execution of a smart contract. However, Ether can also be used as a regular digital currency similar to Bitcoins (BTC) [3].

These blockchain based digital currencies (also called *crypto-currencies*) have a certain anonymous feel to them. Unlike (online) banking, the user does not need to provide any personal details to a central authority to use crypto-currencies. However, the use of crypto-currencies is not as anonymous as is commonly believed, as we will describe later in this paper. This believed anonymity has attracted the attention of those with unlawful intentions. Bitcoin for instance was used to conduct activities such as money laundering and black market trading [4] [5]. These activities make crypto-currency platforms worth investigating from a law-enforcement or forensics point of view.

The aim of this study is to determine if it is feasible to deanonymise Ethereum users by means of the same principles that allowed deanonymisation in the Bitcoin network. In this way we hope to aid law enforcement agencies in tracking and linking activity in the Ethereum blockchain to certain actors that are of interest to them.

We focus on Ethereum due to its prominent position in the crypto-currency arena. At the time of writing (January 2018) Ether has a market capitalisation of over 117 Billion US Dollars and holds the second position after Bitcoin with 223 Billion US Dollars [6]. We chose Ethereum over the even more prominent Bitcoin as there has already been a significant effort in researching the privacy of Bitcoin.

We limit ourselves to investigating attacks on the privacy of users because such attacks are most useful for forensics. Attacks that focus on manipulating the network to one's own benefit are not relevant to our goal of tracking suspect actors within the network. We chose to apply existing attacks on the Bitcoin network to Ethereum, because ample literature exists on attacking Bitcoin and some of these attacks could be viable when applied to Ethereum.

The remainder of this paper is structured as follows: Section II outlines our main research question and our sub-questions. Section III describes the possible ethical implications of our research. In Section IV the related work in this field is described. In Section V the Bitcoin platform and its inner workings are explained while Section VI describes the inner workings of Ethereum. In Section VII the existing attacks on the Bitcoin network are discussed. In Section VIII we will explore the feasibility on Ethereum of the attacks targeting Bitcoin. In Section IX we discuss the results and Section X our conclusions. We conclude our paper with suggestions for future work in Section XI.

II. Research questions

Following from the introduction, our research question is defined as:

- Is deanonymisation of clients feasible for the Ethereum network?

In order to answer our main research question, the following sub-questions are defined:

- What differences exist between the Ethereum and Bitcoin networks?
- What deanonymisation methods exist for the Bitcoin network?
- Is it possible to deanonymise Ethereum clients using the same methods?

III. Ethical implications

In our research we will investigate the privacy of Ethereum users and attempt to find a deanonymisation method that can track Ethereum addresses or link them to IP addresses. Our main focus is a theoretical investigation of existing methods made for the Bitcoin network and if they can be applied to the Ethereum network. We will not put these methods into practice and therefore do not compromise the privacy of any Ethereum or Bitcoin users.

IV. Related work

Biryukov et al. developed a method to deanonymise Bitcoin clients by linking IP addresses to wallet addresses [7]. This attack focuses on the Bitcoin P2P network and relies on the principle that the first peer to broadcast a transaction is likely the creator of that transaction. With sufficient resources Biryukov et al. would manage to deanonymise clients with a success rate ranging from 11% up to 60% in 2014. This attack is relatively efficient compared an attack that connects to all nodes in the P2P network, however it still requires a significant number of connections and is therefore still rather resource intensive. With the increased size of the Bitcoin network at the time of writing, significant additional resources would likely be needed to establish the necessary connections for this deanonymisation method.

Conti et al. present a survey on security and privacy issues of Bitcoin [8]. Most relevant for our research are the several privacy issues covered therein, which show that Bitcoin does not provide anonymity but instead provides only pseudonymity. Even if users create multiple Bitcoin addresses, the addresses belonging to the same user can be found through blockchain analysis, they state. A tool that analyses the Bitcoin blockchain in such a way, BitIodine, has been developed by Spagnuolo et al. [9]. We further discuss BitIodine in Section VII-B. However, these methods of deanonymisation differ from the previous approach as no link is made to IP addresses. Instead, clients are tracked based on their Bitcoin addresses only. BitIodine uses the *change* characteristic of Bitcoin, where excess currency in a transaction is returned to the sender as change, to cluster addresses belonging to the same user.

Al Jawaheri presents a way to deanonymise Bitcoin users by combining posts on social media with clustering methods for Bitcoin addresses [10]. Their method works by first scraping social media platforms for publicly posted Bitcoin addresses, posted for instance with the intent to receive donations. The gathered addresses are then used to find a cluster of addresses belonging to that same user through applying one of the heuristics also used by Spagnuolo et al. With this set of addresses, Al Jawaheri is able to link the user's social media account to transactions that were thought to be anonymous, even if anonymisation tools such as Tor were used for the transaction [10]. This method relies on a user posting at least one of their Bitcoin addresses publicly on social media and the information of their social media profile being accurate to effectively deanonymise them.

Atzei et al. present a survey of the vulnerabilities that exist in Ethereum's smart contracts [11]. Their paper describes several attacks that are aimed at illegitimately obtaining Ethereum's currency, *Ether*. Among the described attacks is the well known attack on the Distributed Autonomous Organisation (DAO) which implements a crowd funding platform for the development of Ethereum. This sparked a hard-fork of the Ethereum blockchain in the summer of 2016 [11]. None of the attacks on smart contracts described in this paper are aimed at invading the user's privacy, which makes the investigation of such attacks a relevant field of study.

V. Bitcoin architecture

Like most blockchain based platforms, Bitcoin consists of two key elements, a peer-to-peer (P2P) network and a blockchain. The P2P network is unstructured and anyone can join using a persistent TCP connection [12]. Joined nodes can add new transactions to this public ledger that is run by the nodes in the network called *miners*. This means there is no need for a central authority in the network, making it fully decentralised [13]. Each element will be discussed in turn below.

A. P2P Network

Through the P2P network of Bitcoin, both new transactions that are being broadcast, and the blockchain consisting of all old transactions are transferred between nodes [14]. A more detailed explanation of the blockchain of Bitcoin can be found in Section V-B.

Each node aims to maintain exactly eight so called *entry nodes*, see Figure 1. The entry nodes are the eight *server* nodes (i.e. nodes that accept incoming connections) to which a Bitcoin client establishes an outbound connection. Server nodes themselves also maintain those eight outgoing connections, i.e. entry nodes. Through these entry nodes the client communicates to the rest of the network. Each node maintains the connection to its entry nodes until they go down, at which point a new entry node is found. Nodes listen by default on port 8333 for inbound connections, but do not necessarily have incoming connections. If a node connects to the network for the first time, it will connect to one of the hard-coded seed servers [15]. DNS queries are done to those seed servers to update the client's address database with new nodes. For every address that is added to the database, a timestamp is stored as well to ensure its freshness.

After an initial set of peers is discovered through the seed servers, a node can discover additional peers by sending a *getaddr* message to one of its current peers. That peer will

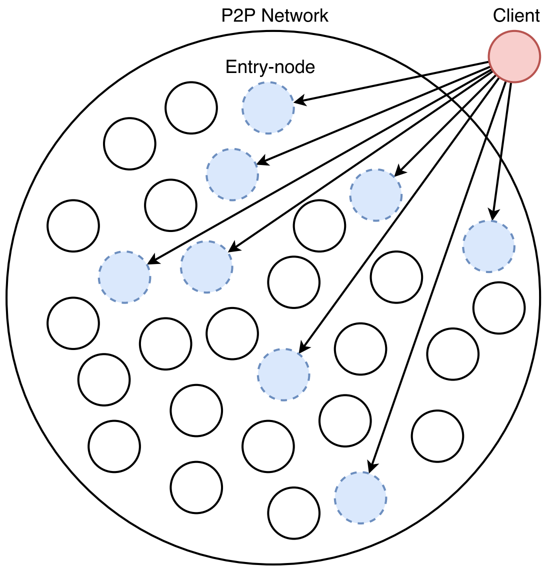

Fig. 1. Entry nodes (blue & dashed line) of a certain client (red) in the Bitcoin P2P network (interconnections not depicted), arrows indicate an outgoing connection from the red client node

then return a list with 23% of its known peers' addresses (selected at random) to the requesting node, with a maximum of 1000 addresses. This list can include any address that the peer knows of, including those that it has no connection to [12] [16].

Transactions propagate through the network from one peer to another in several steps. First the sending peer sends an inv message to its peers to inform them about one or multiple new transaction(s). This message contains a hash of the new transaction(s) [12]. A node that receives the inv message, requests the actual data by sending back a $getdata$ message. The sending peer then responds with the corresponding transaction data. The receiver does several checks on the received data and if those pass, the *trickling* process is started.

Within the transaction propagation, trickling is used as a method to make it harder for adversaries to analyse the traffic and determine which transaction is originating from a certain node. Trickling does this by queuing the inv messages before sending them out [12]. A sending node randomly selects another neighbour from its peer list every 100ms and flushes all outgoing message that are in that neighbour's queue.

Peers remember transactions that have been sent over a connection and will not resent them over the same connection. Such remembered transactions are forgotten eventually if they do not make it into the blockchain [7]. Each node is itself responsible for having its transactions included in the blockchain, so re-sending or re-broadcasting can be necessary when a transaction is still not included in a block after a certain time period [12].

B. Blockchain

Conti et al. describe the Bitcoin blockchain as a "public append-only link-list based data structure which stores the entire network's transaction history in terms of blocks" [8]. Everyone can get a copy of the whole blockhain and see the information that is stored in it. Bitcoin blocks, see Figure 2, consist of a header, transaction counter and transaction list [17] [18] [16].

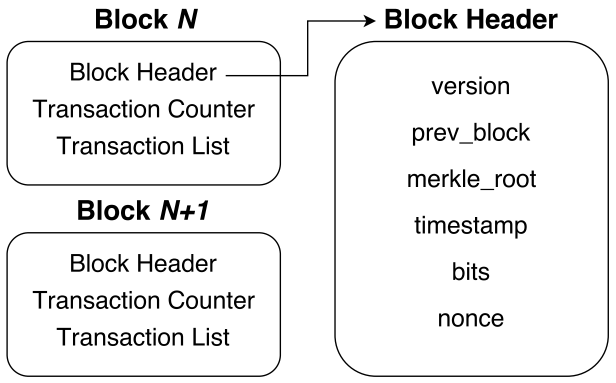

Fig. 2. Overview of the Bitcoin blockchain and the data included in each block

The $prev\_block$ field inside the header is a 32 byte hash of the previous block's header. This hash is calculated by hashing the previous block header twice with the SHA-256 hashing function [19], which creates the link between blocks. The $merkle\_root$ field ensures the integrity of the transactions of the block that it is included in, since it contains a hash based data structure of all transactions in the block.

It could happen that two blocks are mined close to simultaneously. This causes a fork to occur in the blockchain when two or more distinct blocks with the same parent are mined simultaneously. Forks are resolved automatically when ultimately one of the two forks becomes the longer chain and all nodes in the network adopt that chain [20].

The transactions within a block consist of an input and output value [12]. When a user issues a transaction within the Bitcoin network, the output amount of that transaction has to be identical to the input amount. In the case that user A has an input of 3 Bitcoins and wants to transfer only 2 Bitcoins to user B, it means 1 Bitcoin is unused. For this, Bitcoin automatically creates a shadow address that is used to collect the 1 Bitcoin 'change' [9]. Furthermore, the output of a transaction is marked as an *UTXO* (unspent transaction output) [12]. Each UTXO can only be used once as input for a new transaction. After using an UTXO, it is marked as *STXO* (spent transaction output).

C. Consensus Model

Bitcoin relies on a Proof of Work (PoW) consensus model. Any node can connect to the network and start issuing transactions. Those transactions get validated by miners, which bundle a set of valid transactions of their choice and generate a proof of work (PoW) for it [8]. The proof of work consists of a computationally hard puzzle, the difficulty of which depends on the total computational capacity of the network. This PoW is necessary to prevent a so called Sybil attack, where a single party creates a multitude of fake identities in order to manipulate the consensus in the network by creating a false majority [21]. Without PoW, fake identities could write their own blocks which will subsequently be seen as the genuine blocks because the fake identities could have a majority in the network. By requiring a PoW, it becomes difficult to create a fake majority, as significant computational power (more than the entire genuine network) is required. With this model, decisions are thus made not by the majority of nodes, but by the majority of computational power [12].

VI. Ethereum architecture

Just like Bitcoin, Ethereum is a decentralised, blockchain based platform in which users can send and receive the associated currency — Ether in Ethereum's case.

It relies on many of the same principles that Bitcoin does — such as the PoW based consensus model, but its exact workings often differ and are only sparsely documented. An investigation of the Ethereum clients' source code is often necessary to fully understand how Ethereum works. There are also some more obvious key differences between the two. Ethereum's blockchain differs from the one used in Bitcoin as it stores a state, and account balances are stored directly instead of being based on unspent transaction outputs. The P2P network also has significant differences in the way nodes connect to the network and find additional peers. The most notable difference however, is that Ethereum not only allows users to trade Ether but it also provides a blockchain powered application platform that can run so called *smart contracts* [2] [3].

A. Smart Contracts

A smart contract is essentially an Ethereum client with the same permissions and options. However, it is not controlled by a person, but by an arbitrary piece of code written for the Turing complete Ethereum Virtual Machine (EVM). What we call a Smart Contract then is the combination of the Ethereum account and the code running on that account. Each contract can be identified by its Ethereum address in the same way a normal Ethereum user can. Smart contracts are typically not directly written in EVM assembly code, but in the (Ethereum specific) higher level programming language called Solidity [11]. Unlike conventional programs, smart contracts live publicly on Ethereum's blockchain. Any user can create a smart contract, and once it is written and deployed it cannot be altered by anyone, not even by its creator. After deployment, any client can call the contract's functions by sending transactions to it. When a transaction is sent, the corresponding functions from the smart contract are executed by miners and any state-changes are added to the blockchain [11].

Miners are paid for their services with *gas*. Each EVM instruction they execute has an associated gas cost. The inclusion of a gas cost is to prevent DoS attacks on miners through smart contracts that never terminate [11]. The client calling a contract can define the maximum gas that the operation is allowed to consume and the price (in Ether) that they are willing to pay per unit of gas [22].

Smart contracts are a valuable feature of Ethereum. It allows its users to automate many of processes and since contracts are public and immutable once they are deployed, clients using the contract can verify that the contract does what they expect it to.

B. Blockchain

Ethereum blocks, see Figure 3, consist of a block header, a trie data structure containing transactions included in the current block and a list of block headers for the siblings of the block's parent known as *ommers* (i.e. blockchain forks that occurred due to simultaneously mined blocks, one block prior to the current block). The header contains hashes of transaction trie root and the sibling list to prevent tampering after the block has been added to the blockchain. The Proof-of-Work hash for an Ethereum block is computed over the entire header excluding the $mixHash$ and $nonce$ values, which are generated during the PoW computation and added to the header afterwards [23] [3]. It is also noteworthy that, unlike Bitcoin, Ethereum blocks include the state after application of the current block [2].

Ethereum's blocks are thus largely similar to Bitcoin blocks but do include a number of additional values like the previously mentioned list of ommers and transaction receipts which contain the transaction outcomes [3]. Additionally, as opposed to Bitcoin, Ethereum stores more data directly, like account balances and the global state, whereas in the Bitcoin network these things need to be computed. A more detailed description of all values included in Ethereum blocks can be found in the paper by Wood [3].

C. P2P Network

Ethereum's P2P network uses a protocol based on the Kademlia P2P Distributed Hash Table (DHT) [24] but with some significant changes. All DHT functionality is removed, as it is not needed for Ethereum's implementation [25]. Instead, there are six hard-coded bootstrap nodes, called bootnodes, and clients exchange messages to find more peers using a slightly modified version of Kademlia's peer discovery protocol. Unlike in the Bitcoin P2P network, Ethereum's P2P protocol uses $nodeIDs$ to identify nodes. A $nodeID$ is simply a node's public key and is also used to encrypt traffic between nodes. $nodeIDs$ are used in combination with the IP address belonging to that node, which means a link between the two is easily established [24]. To find the exact workings of Ethereum's P2P network, we had to analyse the Ethereum source code as published on GitHub.

Each node keeps track of peers, as per the Kademlia protocol, in rows, where each row i contains peers whose $nodeID$ has the same first i bits as the node itself [25]. For each row the node aims to maintain exactly and stores at most k peers in that row. The value for k can be chosen to determine the level of redundancy in the network [26].

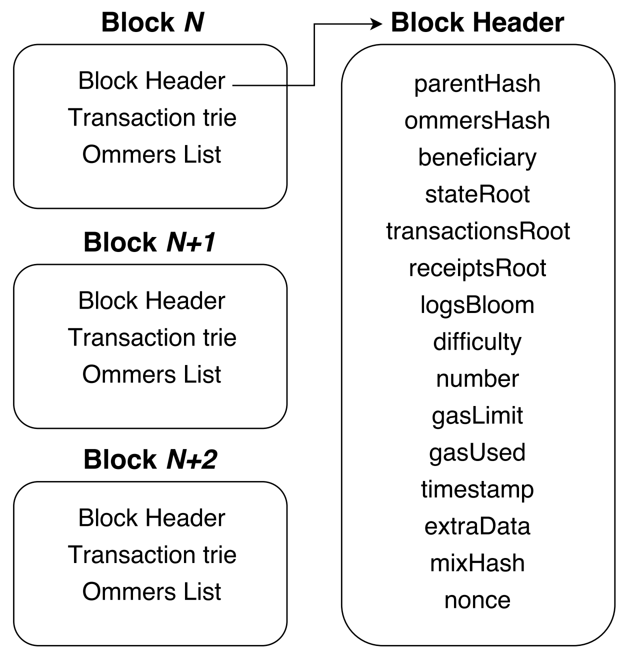

Fig. 3. Overview of the Ethereum blockchain and the data included in each block

A client discovers more peers using a $findnode$ query with its own $nodeID$ as parameter to some of its existing peers [27] [28] [25] [29]. The recipient node of this query will then use the received $nodeID$ to select to which of its peers it will forward the querying node [30] [31] [32]. This is decided by performing an XOR on the SHA3 hashes of the sender's $nodeID$ and each of the $nodeIDs$ of the known peers for this node. The resulting values are sorted and from the addresses of the peers corresponding to the topmost results, known as the *closest* peers, 16 are forwarded to the querying node [29]. The querying node then recursively queries the newly discovered peers to find more peers until no new peers are discovered that are closer to the provided $nodeID$ [26] [24].

To find more peers in this way, a node first needs some entry point into the network to send a $findnode$ query to. For this purpose, currently a total of six hard-coded bootstrap nodes [33], or bootnodes, exist to which a peer can send an *add_me* message to receive a list of peers that are inside the network [33] [26].

VII. Existing attacks

Various attacks have been researched for the Bitcoin network, part of which are aimed at attacking Bitcoin users' privacy by for instance tracking all transactions by a certain user, revealing a user's IPs or mapping which Bitcoin wallets belong to the same person. This Section focuses on two attacks on Bitcoin users' privacy, the first attack aims to link IP addresses to Bitcoin addresses and the second attack tries to cluster different Bitcoin addresses belonging to the same user.

A. Discovering IP Addresses

As mentioned before, Biryukov et al. developed an effective but invasive and resource intensive deanonymisation method for the Bitcoin P2P network. This attack works by identifying as many entry nodes as possible of a client or IP address we are interested in.

Entry nodes are identified by connecting adversary controlled nodes to nearly every Bitcoin server node. These nodes then listen for client IP addresses that are advertised in the network and log the set of server nodes which advertised that address. This set then includes (a subset of) the entry nodes of that specific client but can include some false entries as well. To filter out these false entries Biryukov et al. exploit the fact that Bitcoin clients will not forward an address over a connection if that connection was recently used to forward that same address [7]. Once a subset of the entry nodes is known, the adversary establishes multiple connections to a large amount of the server nodes in the network and listens for transactions that are being broadcast. If a transaction is first broadcast from exactly those nodes that form an entry-node subset, we know that the creator of that transaction is likely the node belonging to those entry nodes.

It is important to note that this attack relies on the fact that entry nodes remain largely constant for a given client. If entry-nodes are not constant, at least to some extent, this attack will no longer be able to effectively deanonymise.

Furthermore, this attack will require significant resources to execute, as a large amount of connections need to be made to nearly all the server nodes in the network. At the time of writing (May 2014) Biryukov et al. applied a proof of concept of this attack on the testnet of Bitcoin which consisted of an average of 240 Bitcoin server nodes. They were able to correctly link 59.9% transactions to the corresponding IP address by adding 50 connections to all server nodes, which results in using 40% of the total connection capacity of those server nodes. This results in a total of 12000 connections within the test network, making this attack on Bitcoin unfeasible or even impossible for the average individual. A large corporation or government agency would be more capable of amassing the required resources and executing such an attack. However, with the increase in size of the Bitcoin network since 2014, which at the time of writing consists of just under 120000 nodes [34] [35], it is possible that by now this type of attack is out of reach for even such parties.

B. Clustering Bitcoin Addresses

Another method for deanonymisation is finding out which Bitcoin addresses belong to the same user. Spagnuolo et al. present a framework that is able to find such a clustering of addresses called BitIodine [9].

BitIodine consists of several modules, of which the three most interesting modules are the scraper, the block parser and the clusteriser. The scraper is used to crawl the web for Bitcoin addresses that users have posted publicly, similar to the techniques described by Al Jawaheri. The block parser stores all Bitcoin blocks into a database and the clusteriser groups addresses using two heuristics.

The first heuristic exploits so called *multi-input* transactions. A multi-input transaction occurs when a user, A, wants to transfer a certain amount of Bitcoins to another user, B, but does not have a single UTXO with enough coins for the transaction. A will then have to use multiple inputs to aggregate the amount A wants to transfer. When a transaction includes multiple input addresses, Spagnuolo et al. assume that those input addresses belong to the same wallet and, in turn, the same user.

The second heuristic uses the concept of change that the Bitcoin network has. As explained in Section V-B, most transactions include change being sent back to the user that the transaction originates from. This fact can be used to identify additional addresses belonging to that user, Spagnuolo et al. use this in BitIodine. They analyse the transactions that contain only two output addresses. If one of the two output addresses never appeared in the blockchain before, it is likely that this address is the newly generated shadow address of the sending user [9].

An adversary with average computational power would be able to apply both the heuristics described previously, and thus make deanonymisation in this way feasible. However, more computational power will result in faster processing of transactions. At the time of writing, the total size of the Bitcoin blockchain is just over 155GB [36]. A government agency or large company potentially has access to much larger the computational power, and could therefore speed up processing of all this data significantly.

VIII. Feasibility of existing attacks for Ethereum

When applying the first attack described by Biryukov et al. to Ethereum's P2P network, we encounter some issues. Ethereum clients do not have certain static entry nodes and the connections to their peers are more volatile — if a closer peer is found it will connect to that peer and possibly disconnect from a previous peer. Additionally, Ethereum clients may connect to many more than eight peers. The exact number is defined by each client and is not a global setting. Monitoring all nodes that a certain client is connected to by finding a super-set of entry nodes and then narrowing this set down, therefore becomes troublesome. This attack is therefore not feasible for Ethereum in its current form. If a different method for identifying those nodes is adopted, this strategy of identifying the nodes to which a client is connected and then using that to deanonymise that client, can be feasible for Ethereum. An adversary could attempt identifying or taking control over the nodes to which a client is connected by exploiting Ethereum clients' nature to connect to nodes whose public key hashes close to their own.

Another issue with the attack described by Biryukov et al. is the scale of the Ethereum network, which at the time of writing consists of over 24000 nodes [37]. An adversary that wishes to deanonymise Ethereum users by finding an IP address when given an Ethereum address, will need to monitor the entire network by connecting to practically every node. Due to the size of the network, this approach requires significant resources. However, if an adversary does have access to the necessary resources, this attack will be effective. An adversary whose goal is to identify a wallet address when given a certain IP address will require less resources to do so. In this case, the adversary can find the node with the given IP address in the P2P network and attempt to monitor all its connections. This approach is less impacted by the scale of the network and depends only on the number of peers that the certain node is connected to. This number is also likely to increase with the network size, as a larger network leads to each node being able to fill up more of its rows with peers. However, the increase in connected peers will be much lower than the total increase in network size.

When we look at the second attack on users' privacy described by Spagnuolo et al. and attempt to apply its clustering methods to the Ethereum blockchain, issues arise as well. As described previously, Ethereum does not work with UTXOs but instead stores a state that keeps track of which funds belong to what address. For this reason, multi-input transactions do not exist in the same form as they do in the Bitcoin network. This means that clustering addresses based on multi-input transactions is not possible for Ethereum.

Due to the lack of UTXOs, Ethereum also does not need the concept of change that Bitcoin does. A user can simply transfer exactly the desired amount of funds to another user without having to 'use up' an entire UTXO. Clustering addresses based on guessing shadow addresses through the change feature therefore also becomes impossible, meaning that both clustering techniques are not applicable to the Ethereum network.

However, the scraping functionality in BitIodine can be successfully applied to Ethereum, as Ethereum users might also post their addresses online publicly.

IX. Discussion

Both the discussed Bitcoin deanonymisation attacks are difficult to apply to the Ethereum network. In the case of the attack by Biryukov et al. the specifics of the Bitcoin P2P network that the attack relies on do not exist in the Ethereum network. Ethereum's network however, does have other features that can potentially be exploited by an attacker, such as the peer selection protocol which chooses peers based on their public key's hash closeness.

The clustering methods described by Spagnuolo et al. are also difficult to apply, as they rely on Bitcoin's UTXO based system which does not exist in Ethereum. Clustering addresses in Ethereum is also less necessary, as users have no reason to create additional addresses due to the lack of a change concept in Ethereum. If a user does create multiple Ethereum addresses, clustering or tracking those will require similar techniques to those that are used when finding relations between regular bank accounts, which are out of scope for this paper.

Other possibilities for attacks on users' privacy exist but are not covered in this paper, such as attacking the peer selection protocol, the bootnodes or the wallet software. We briefly discuss the possibilities for such attacks in Section XI.

X. Conclusions

Due to the differences between the Bitcoin and Ethereum networks, deanonymisation of Ethereum clients turned out to be unfeasible using existing methods designed for the Bitcoin network. Both of the investigated methods rely on specific features in the Bitcoin network. The IP discovery attack relies on the static entry nodes that exist in Bitcoin's P2P network and the clustering attack uses the UTXO based transactions to cluster addresses, both of which do not exist in the same way in Ethereum.

However, Ethereum does have other potentially exploitable features such as the hard-coded bootnodes and the peer selection protocol. Furthermore, Ethereum does not facilitate the creation of shadow addresses that exist in Bitcoin, potentially making users traceable without clustering metrics, as in most cases they only use a single address.

XI. Future work

An attack which builds on the same principle of monitoring the nodes to which a certain client is connected, might be possible by creating a shadow P2P network and taking control of Ethereum's bootnodes. An adversary with sufficient resources could create a shadow P2P network which connects to the real network but does not reveal any legitimate nodes to its peers. If the adversary also gains control over all bootnodes and modifies them to point newly joining nodes only to the adversary's nodes, a situation arises where the adversary controls all peers of a newly joined node. This enables her to find which Ethereum address belongs to that node when it broadcasts a transaction. Since IP addresses of nodes are known, it allows the adversary to link this IP address to the Ethereum address used for the transaction. This kind of attack would not be feasible for an average person, but a government agency could forcibly take control over bootnodes to mount this kind of attack. We have not implemented or verified the effectiveness of such an attack and leave it as a possibility for future work.

Another method to achieve control over all of a node's peers and deanonymise the user running that node could be to abuse the peer selection protocol. An Ethereum node selects its peers based on their closeness to the node itself, which is determined by doing an XOR of the SHA3 hash of both nodes' public keys. This trait can be exploited by attempting to create nodes with public keys that hash close to a certain target node. Generating and hashing public keys is not an exceptionally expensive operation, making this attack feasible when dealing with a sparse network. One method to prevent such an attack is to assign the public keys instead of letting users generate them themselves. This method has also been applied to DHTs using smartcards to distribute the key material Druschel and Rowstron [38]. This would have the negative effect of introducing a centralised element into a system that aims to be fully decentralised and is therefore unlikely to see adoption by Ethereum. Additionally, even if an adversary manages to generate multiple public keys that hash close to the target node, they have no way to know for certain what other nodes the target node might be connected to, thus reducing the success rate of this kind of attack.

Another approach could be not to gain control over, but only to identify the nodes that a certain client is connected to. When such nodes are identified, listening for transaction broadcasts in the same way that Biryukov et al. do may allow deanonymisation of clients. The adversary could attempt this by hashing the public keys of all nodes it can find and determining which nodes the client they are interested in is likely to connect to. However, this approach gives no assurance that those nodes are the right nodes. This paper has not explored the possibilities for such an attack but we think it warrants further investigation and as such leave it as a suggestion for future work.

Another possibility for deanonymisation in the Ethereum network exists through an attack on wallet software that clients use. If intelligence agencies find a vulnerability in existing wallet software or create their own and manage to achieve widespread adoption, tracking users based on IP or Bitcoin address becomes viable without having to expend significant resources. Such investigations are out of scope for this paper, so again leave this as a suggestion for future work.

References

[1] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. 2008. URL: https://bitcoin.org/bitcoin.pdf.

[2] Vitalik Buterin et al. *Ethereum white paper*. 2013. URL: https://github.com/ethereum/wiki/wiki/White-Paper.

[3] Gavin Wood. "Ethereum: A secure decentralised generalised transaction ledger". In: *Ethereum Project Yellow Paper* 151 (2014). URL: https://ethereum.github.io/yellowpaper/paper.pdf.

[4] Malte Moser, Rainer Bohme, and Dominic Breuker. "An inquiry into money laundering tools in the Bitcoin ecosystem". In: *eCrime Researchers Summit (eCRS), 2013*. IEEE. 2013, pp. 1–14.

[5] Nicolas Christin. "Traveling the Silk Road: A measurement analysis of a large anonymous online marketplace". In: *Proceedings of the 22nd international conference on World Wide Web*. ACM. 2013, pp. 213–224.

[6] CoinMarketCap. *Cryptocurrency Market Capitalizations*. URL: https://coinmarketcap.com/ (visited on 01/11/2018).

[7] Alex Biryukov, Dmitry Khovratovich, and Ivan Pustogarov. "Deanonymisation of clients in Bitcoin P2P network". In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM. 2014, pp. 15–29.

[8] Mauro Conti, Chhagan Lal, Sushmita Ruj, et al. "A Survey on Security and Privacy Issues of Bitcoin". In: *arXiv preprint arXiv:1706.00916* (2017).

[9] Michele Spagnuolo, Federico Maggi, and Stefano Zanero. "Bitiodine: Extracting intelligence from the bitcoin network". In: *International Conference on Financial Cryptography and Data Security*. Springer. 2014, pp. 457–468.

[10] Husam Basil Al Jawaheri. "Deanonymizing tor hidden service users through bitcoin transactions analysis". MA thesis. 2017. DOI: http://hdl.handle.net/10576/5797.

[11] Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli. "A Survey of Attacks on Ethereum Smart Contracts (SoK)". In: *International Conference on Principles of Security and Trust*. Springer. 2017, pp. 164–186.

[12] Florian Tschorsch and Björn Scheuermann. "Bitcoin and beyond: A technical survey on decentralized digital currencies". In: *IEEE Communications Surveys & Tutorials* 18.3 (2016), pp. 2084–2123.

[13] Sarah Meiklejohn et al. "A fistful of bitcoins: characterizing payments among men with no names". In: *Proceedings of the 2013 conference on Internet measurement conference*. ACM. 2013, pp. 127–140.

[14] Joan Antoni Donet Donet, Cristina Pérez-Sola, and Jordi Herrera-Joancomartí. "The bitcoin P2P network". In: *International Conference on Financial Cryptography and Data Security*. Springer. 2014, pp. 87–102.

[15] Bitcoin Core Team. *Bitcoin Core - Source code*. 2018. URL: https://github.com/bitcoin/bitcoin/blob/595a7bab23bc21049526229054ea1fff1a29c0bf/src/chainparams.cpp#L128 (visited on 01/22/2018).

[16] Bitcoin.org. *Bitcoin Developer Reference*. 2018. URL: https://bitcoin.org/en/developer-reference (visited on 01/24/2018).

[17] John Ratcliff. *How to Parse the Bitcoin BlockChain*. 2014. URL: http://codesuppository.blogspot.nl/2014/01/how-to-parse-bitcoin-blockchain.html (visited on 01/24/2018).

[18] Bitcoin Wiki. *Protocol documentation*. 2018. URL: https://en.bitcoin.it/wiki/Protocol_documentation (visited on 01/24/2018).

[19] Kiran Vaidya. *Bitcoin's implementation of Blockchain*. 2016. URL: https://medium.com/all-things-ledger/bitcoins-implementation-of-blockchain-2be713f662c2 (visited on 01/24/2018).

[20] Cuneyt Gurcan Akcora, Yulia R Gel, and Murat Kantarcioglu. "Blockchain: A Graph Primer". In: *arXiv preprint arXiv:1708.08749* (2017). URL: https://arxiv.org/abs/1708.08749.

[21] Arati Baliga. *Understanding blockchain consensus models*. Tech. rep. Persistent Systems Ltd, 2017.

[22] Ethereum Community. *go-ethereum transaction.go*. 2017. URL: https://github.com/ethereum/go-ethereum/blob/6f69cdd109b1dd692b8dfb15e7c53d2051fbc946/core/types/transaction.go (visited on 01/24/2018).

[23] Ethereum Community. *Ethash Wiki*. 2017. URL: https://github.com/ethereum/wiki/wiki/Ethash (visited on 01/24/2018).

[24] Petar Maymounkov and David Mazieres. "Kademlia: A peer-to-peer information system based on the xor metric". In: *International Workshop on Peer-to-Peer Systems*. Springer. 2002, pp. 53–65.

[25] Ethereum Community. *RLPx: Cryptographic Network & Transport Protocol*. 2015. URL: https://github.com/ethereum/devp2p/blob/master/rlpx.md (visited on 01/24/2018).

[26] Ethereum Community. *Kademlia Peer Selection*. 2015. URL: https://github.com/ethereum/wiki/wiki/Kademlia-Peer-Selection (visited on 01/24/2018).

[27] Ethereum Community. *go-ethereum udp.go*. 2017. URL: https://github.com/ethereum/go-ethereum/blob/1d06e41f04d75c31334c455063e9ec7b4136bf23/p2p/discover/udp.go (visited on 01/24/2018).

[28] Adem Efe Gencer et al. "Decentralization in Bitcoin and Ethereum Networks". In: *arXiv preprint arXiv:1801.03998* (2018). URL: https://arxiv.org/abs/1801.03998.

[29] Luke Anderson et al. "New kids on the block: an analysis of modern blockchains". In: *arXiv preprint arXiv:1606.06530* (2016). URL: https://arxiv.org/abs/1606.06530.

[30] Ethereum Community. *go-ethereum hive.go*. 2017. URL: https://github.com/ethereum/go-ethereum/blob/32516c768ec09e2a71cab5983d2c8b8ae5d92fc7/swarm/network/hive.go (visited on 01/24/2018).

[31] Ethereum Community. *go-ethereum kademlia.go*. 2017. URL: https://github.com/ethereum/go-ethereum/blob/32516c768ec09e2a71cab5983d2c8b8ae5d92fc7/swarm/network/kademlia/kademlia.go (visited on 01/24/2018).

[32] Ethereum Community. *go-ethereum address.go*. 2017. URL: https://github.com/ethereum/go-ethereum/blob/86f6568f6618945b19057553ec32690d723da982/swarm/network/kademlia/address.go (visited on 01/24/2018).

[33] Ethereum Community. *go-ethereum bootnodes.go*. 2018. URL: https://github.com/ethereum/go-ethereum/blob/c335821479db9930a98cbd48996f880c35a59797/params/bootnodes.go#L21 (visited on 01/24/2018).

[34] Addy Yeow. *Global Bitcoin Node Distribution*. 2017. URL: https://bitnodes.earn.com/ (visited on 02/07/2018).

[35] Blockchain.info. *Bitcoin Charts & Graphs*. 2018. URL: https://blockchain.info/charts (visited on 02/01/2018).

[36] Blockchain.info. *Blockchain Size*. 2018. URL: https://blockchain.info/nl/charts/blocks-size (visited on 02/07/2018).

[37] EtherNodes. *EtherNodes - The Ethereum Node Explorer*. 2017. URL: https://www.ethernodes.org/ (visited on 02/07/2018).

[38] Peter Druschel and Antony Rowstron. "PAST: A large-scale, persistent peer-to-peer storage utility". In: *Hot Topics in Operating Systems, 2001. Proceedings of the Eighth Workshop on*. IEEE. 2001, pp. 75–80.