2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)

Practical Deanonymization Attack in Ethereum Based on P2P Network Analysis

Yue Gao*†‡, Jinqiao Shi*§, Xuebin Wang*†‡, Ruisheng Shi§, Zelin Yin*† and Yanyan Yang¶
* Institute of Information Engineering Chinese Academy of Sciences
† National Engineering Laboratory for Information Security Technologies
‡ School of Cyber Security, University of Chinese Academy of Sciences
§ Beijing University of Post and Telecommunications
¶ Department of Information Technology and Cyber Security, People's Public Security University of China

***Abstract*—Ethereum is the second-largest cryptocurrency, which is an open-source public blockchain platform with smart contract functionality. With the increasing popularity of Ethereum, considerable attention has been paid to its privacy and anonymity. Previous work in Ethereum deanonymization mostly focused on the analysis of its transaction graph and user behaviors. In this paper, for the first time we explored the feasibility of deanonymizing Ethereum users based on P2P network analysis. By measurement and analysis, we observed that the attacker can make connections with approximately 90% mainnet synced full nodes. Based on the well-connected supernode, the deanonymization experiments with basic estimators preliminarily indicate that the anonymity of Ethereum P2P network is pretty limited. To further improve the effect of deanonymization, we implemented and evaluated a machine learning based estimator, which reduces the influence of network delay on deanonymization and thus increases the success rate to 88%. At last, we provide the discussion about the anonymity and efficiency of the propagation mechanisms.**

***Index Terms*—Ethereum, Blockchain, Anonymity, P2P network**

I. Introduction

Ethereum has grown rapidly since its inception in 2015. And Ether is the native cryptocurrency of the platform. Ranking after Bitcoin, it is the second-largest cryptocurrency by market capitalization. In the circulation of Ether, the identities of users are hidden behind the addresses generated from public keys. To some extent, the pseudonym mechanism protects the privacy of users. However, it is known that these pseudo-anonymous transactions are also abused in illegal activities where the criminals try to obscure their money traces, like money laundering, drug dealing and so on.

The anonymity and privacy of cryptocurrencies, like Bitcoin and Ethereum, have attracted accumulating attention and research. Some studies try to enhance the anonymity of cryptocurrencies [13]–[17], while others manage to deanonymize users from various perspectives. The deanonymization attacks can be divided into two categories, aiming at identifying addresses that belong to the same user [5], [6], [18], [19], or revealing the relationship between addresses and real-world identities, such as IP addresses. The former is necessary for deanonymization, because criminals may create large numbers of addresses to make transactions for hiding themselves better. Besides, normal users may also create new addresses with the consideration of privacy. But to achieve the ultimate goal of deanonymization, a further step needs to be taken to link these addresses to real-world identities. One way to map addresses and thus entities to identities is by gathering information from side channels. For example, addresses exposed in news reports or publicly declared by users on social networks. A more rigorous way is exploiting the information revealed on the P2P network to correlate transactions to their originators' IP addresses.

Corresponding author: Xuebin Wang, Email:wangxuebin@iie.ac.cn

Previous research on revealing users' identities based on P2P network analysis was mainly conducted on Bitcoin [8]–[10]. In this paper, we will focus on Ethereum, which has a different network structure and protocols. The deanonymization attack on blockchain P2P network was first proposed by Dan Kaminsky during the 2011 Black Hat conference [2], which has been applied [7] and theoretically analyzed [9] in Bitcoin. This kind of attack relies on a supernode that connects to all nodes in the network and listens to the relayed transactions. Then the attacker infers the source node based on the estimator with a certain strategy. In this paper, we manage to answer two involved questions in the practical attack. One is what coverage of Ethereum nodes the attacker can make connections with, and the other is which strategy should be adopted in Ethereum to infer the source node accurately.

Our experiments show that there is around 10K synced full nodes in Ethereum mainnet. The supernode can maintain connections with about 90% of them. The main reason for disconnection is the instability of the network. By analyzing the communication protocol and its implementation, we found that Ethereum nodes send transactions to their neighbors immediately, i.e., without any extra random waiting time. In this case the main factor affecting the deanonymization is network delay. There are two basic estimators to infer the source node. The firstReach estimator outputs the node whose transaction message arrives the supernode first as the source node. However, the triangle inequality violations of network delay may lead to the failure of the deanonymization. For the firstSend estimator, it outputs the node to send out the

978-0-7381-2646-3/21/$31.00 ©2021 IEEE
DOI 10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00191

transaction earliest as the source node. The time when the transaction was sent is estimated by measuring the delay between the supernode and the senders. However, the errors of delay estimation may cause the failure of the deanonymization. To combine the two estimators and thus infer the source node with a higher success rate, we propose a machine learning based estimator. Our experiments on Ethereum mainnet show that the ML-based estimator can deanonymize transactions with 88% success rate, which is 8% and 10% higher than that of the firstReach estimator and the firstSent estimator.

The following of this paper is structured as follows: Section II briefly introduces the background knowledge of Ethereum, especially its P2P network; Section III gives the definition of the deanonymization problem and the analysis of two key questions for the practical attack; In Section IV we provide the experiment results of the deanonymization experiments on Ethereum testnet and mainnet, including the comparison between basic estimators and proposed ML-based estimators; In Section V we give some further discussion about the propagation mechanisms. Section VI reviews related work; Finally, we conclude our paper in Section VII.

II. Background

A. Ethereum P2P network

Ethereum is managed by a fully distributed structured P2P network. The network communication is composed of a set of protocols, namely Devp2p protocol [1]. Devp2p protocol is the main component of common Ethereum clients, like Geth (the official Go client) and OpenEthereum (an unofficial Rust client). The protocol suite includes several bottom protocols: Ethereum Node Records, Node Discovery Protocol and RLPx transport protocol. It also defines RLPx-based application-level protocols, such as Ethereum Wire Protocol.

- **Ethereum Node Records** is an open data format for P2P connections. Node record consists of three parts, signature, sequence number and the key/value pairs of node information. The last part usually contains IP, port and so on, for other nodes to decide whether to connect to the node.
- **Node Discovery Protocol** is based on Kademlia DHT for the storage and retrieval of Ethereum nodes. Each node has a cryptographic identity. The public key serves as its nodeID, and the private key is used to sign transactions. The logical distance between two nodes is decided by XOR operation of nodeID hashes.
  Ethereum node discovery is implemented via UDP protocol, including three pairs of messages. 1) *Ping* and *Pong*: to detect the node status; 2) *FindNode* and *Neighbors*: to query nodes closest to the target; 3) *EnrRequest* and *EnrResponse*: to request the node record. Besides, the Geth team maintains a list of reachable nodes by periodically traversing the DHT. It is a supplement to hardcoded bootstrap nodes and traditional node discovery protocol.
- **RLPx Transport Protocol** is a TCP-based protocol for information exchange between nodes. The establishment of connection includes key exchange and protocol handshake. Key exchange is based Diffie Hellman algorithm for further encrypted communication. Protocol handshake is implemented by exchanging *Hello* messages, which contains protocol version, clientID, capabilities, listening port and nodeID. The capabilities represent the supported application-level protocol. For example, eth/65 indicates that the node supports v65 Ethereum Wire Protocol.
- **Application-level Protocols** for Ethereum includes Ethereum Wire Protocol (*eth*), Light Ethereum Subprotocol (*les*), Parity Light Protocol (*pip*) and so on. The *eth* protocol is the main protocol in the current Ethereum network. When two nodes complete the RLPx handshake, they need to exchange the *Status* message first. The *Status* message contains protocol version, networkID, difficulty, current block hash, genesis block hash and forkID. The networkID and genesis hash of Ethereum mainnet is 1 and 0xd4e5...8fa3. The latest fork is the Berlin hard fork on April 15, 2021.
  Nodes that support the *eth* protocol are considered to be Ethereum full nodes, which synchronize blockchain data through the relay of transactions and blocks. There are two protocols for light nodes, *les* and *pip*. Light nodes only download the block header and query other information on demand. Light nodes can create transactions, but they don't participate in the relay of transactions and blocks. Some full nodes also support light protocols to provide services for light nodes.

B. Address and Transaction

- **Address** in Ethereum can either be an externally owned account (EOA) address, or a smart contract account address. The EOA address is generated by secp256k1 elliptic curve encryption algorithm, controlled by the private key. The smart contract account address is determined by the sender's address and the number of its generated transactions (nonce), controlled by the contract code.
- **Transaction** is a piece of data initiated by the signature of the EOA address. All valid transactions will be included in the blockchain data. There are three kinds of transactions: 1) Normal Transactions: sent from an EOA address to the other. This is the most common kind of transaction. The transactions transfer ETH between EOA addresses; 2) Contract Deploying Transactions: sent from an EOA address to the zero-account. The purpose of these transactions is to deploy smart contracts. As mentioned above, the contract address will be determined by the sender and its nonce; 3) Contract Executing Transactions: sent from an EOA address to a deployed contract address. It can be seen that all Ethereum transactions are triggered by EOA addresses.

III. Deanonymization on P2P network

Ethereum provides privacy through the pseudonym mechanism. The identities of Ethereum users are hidden behind

addresses generated from public keys. From the perspective of its P2P network, transactions are broadcast to the whole network. Each Ethereum node can be either the creator of the transaction or just the forwarder. The deanonymization in the P2P network is to identify the IP address of the transaction creator, i.e., the source node. The P2P network of Ethereum nodes can be modeled as a graph $G(V, E)$, where V is the set of all nodes and E is the set of edges or connections between them. The goal of deanonymization is to identify the source node $v^* \in V$.

Deanonymization on the P2P network of cryptocurrency was first proposed by Dan Kaminsky at the 2011 Black Hat conference. As shown in Fig.1, the author pointed out that when the supernode controlled by the attacker has established connections with all nodes in the network, he listens to all relayed transactions and concludes that the first node that announces the transaction may be the source node.

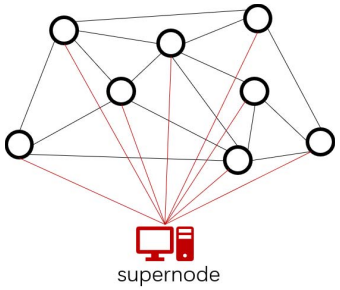


Fig. 1. Deanonymization on P2P network.

Inspired by the technique, in this paper we will study the feasibility of deanonymization in the real-world Ethereum network. In practice, the deanonymization attack can be divided into two phases, connecting to all Ethereum nodes and then inferring the source node of the transactions.

A. Connect to Ethereum nodes

Ethereum nodes can be divided into reachable nodes and unreachable nodes, according to whether incoming connections are accepted. For the latest version of Geth, each node is allowed to establish up to 8 outgoing connections to other nodes and maintains up to 50 active connections in total. While for the latest version of OpenEthereum, each reachable node accepts up to 25 incoming connections and maintains up to 50 total active connections. The supernode cannot establish the connections actively with unreachable nodes behind NATs or firewalls. For these unreachable nodes, the only thing the supernode can do is to listen and wait for their incoming connections. Besides, for reachable nodes, the connection requests from the supernode are not always accepted by nodes due to the limit of the maximum number of connections.

EthNodeFinder. To obtain the view of the real-world Ethereum network, we built ethNodeFinder to collect active Ethereum nodes. EthNodeFinder is based on the modified Geth client (v1.10.1) with differences in three aspects: Firstly, ethNodeFinder logs HELLO, DISCONNECT and STATUS messages to record node information, and TRANSACTIONs and NEW_POOLED_TRANSACTION_HASHES messages to verify whether nodes are completely synced. Secondly, ethNodeFinder disconnects from nodes within one minute to reduce resource consumption. Thirdly, ethNodeFinder periodically reconnects known nodes who sent DISCONNECT with reason *too many peers*.

EthTXListener. To obtain the full visibility of the transaction propagation, we built ethTXListener to maintain connections with as many Ethereum nodes as possible. EthTxListener differs from the unmodified Geth client (v1.10.1) in four aspects: Firstly, ethTxListener ignores the maximum peer connections limit to accept more connections from Ethereum nodes. Secondly, ethTxListener logs TRANSACTIONs and NEW_POOLED_TRANSACTION_HASHES messages, which include nodeID, nodeIP, TxHash and arrival timestamp. Thirdly, ethTxListener doesn't send transactions out, and doesn't synchronize block data either. Fourthly, ethTxListener only maintains connections with completely synced Ethereum nodes, since unsynced nodes can't create or relay transactions.

B. Infer the source node

The strategy to infer the source node depends on the analysis of the transaction propagation. Although most types of Ethereum clients are based on Devp2p protocol, the implementation may be slightly different: 1) Naive-Flooding: For early Geth clients, nodes send the transaction to all neighbors; 2) Semi-Flooding: To reduce the traffic burden, current Geth clients select $\sqrt{n}$ neighbors to send the complete transaction, and then only announce the transaction hash to the other neighbors; 3) Gossip: OpenEthereum clients select $\sqrt{n}$ neighbors to send the complete transaction, and send nothing to the other neighbors. Even if the supernode has established a connection with an OpenEthereum node, it may not be chosen by the node to send transactions to. This is the main difference between Geth and OpenEthereum in terms of propagation mechanism. Through the case study of Geth and OpenEthereum, it can be found that although their propagation mechanisms are not the same, they all forward transactions immediately. Compared with Bitcoin, there is no extra random waiting time before sending transactions, thus the main factor affecting the deanonymization of Ethereum is network delay.

FirstReach Estimator. The method proposed by Dan Kaminsky [2] to infer the source node can be called FirstReach estimator. Let τ_v denote the time the supernode receives the transaction from node $v \in V$, and τ denote the set of all arrival timestamps. The FirstReach estimator M_{FR} outputs the first node to report the transaction to the supernode as:

$$M_{FR}(\tau, G) = \underset{v \in V}{arg\,min}\,(\tau_v). \tag{1}$$

The success of the FirstReach estimator is based on the assumption that the network delay time of one hop forwarding is always less than that of two hops. However, many studies [3], [4] have shown that triangle inequality violation (TIV) is

a persistent and widespread phenomenon in the network. As shown in Fig.2, the transaction is created by A. The arrival timestamp of the transaction sent directly from A to C is τ_A, while the arrival timestamp of the transaction forwarded by A to B and then from B to C is τ_B. The network delay between A and C, A and B, B and C are denoted as $delay_{AC}$, $delay_{AB}$, $delay_{BC}$, respectively. If $delay_{AC} \geq delay_{AB} + delay_{BC}$, the supernode will receive the transaction forwarded by other nodes earlier than the source node, i.e., $\tau_A \geq \tau_B$.

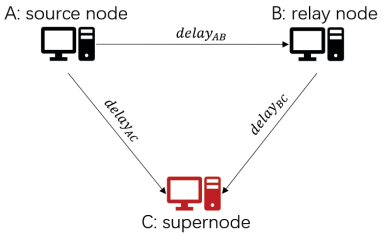


Fig. 2. Triangle Inequality Violations (TIVs) in network.

FirstSent Estimator. The intuitive solution to triangle inequality violations is to measure the network delay between the supernode and other nodes. Let δ_v denote the measured network delay between the supernode and the node $v \in V$, δ denote the set of delays. Then the attacker infers the source node according to estimated sending timestamp as:

$$M_{FS}(\tau, \delta, G) = \underset{v \in V}{arg\,min}\,(\tau_v - \delta_v). \tag{2}$$

The performance of the FirstSent estimator is closely related to the delay estimation. However, the delay estimation cannot be absolutely accurate. For example, we can only obtain the RTT delay, but the asymmetry may lead to the error of estimation.

ML-based Estimator. The two basic estimators have their own shortcomings, in some cases the FirstReach estimator can correctly infer the source node, while in other cases the FirstSent estimator performs better. Inspired by this, we expect to train a machine learning based estimator to combine the two estimators, so as to infer the source node with a higher success rate without human decision.

The problem of deanonymization can be regarded as a multiclass classification problem. Each node can be considered as a class. However, the traditional multiclass classifier is not suitable for this problem, since the characteristics of each node (class) are changed for different transactions. Therefore, we first simplify the problem to a binary classification problem, that is, to judge whether a transaction is sent by a target node. Then, for the same transaction, the node with the largest positive probability output by the binary classifier is considered as the source node. In Section IV, we provide details of the combined estimators and the deanonymization success rate of the three estimators.

IV. Experiments and Evaluation

A. The bird's-eye view of Ethereum P2P network

Experiment Setup. From June 18th-July 2nd, 2021, we ran 40 instances of ethNodeFinder on 20 Ubuntu 16.04 machines with 2GB RAM and 1 core CPU.

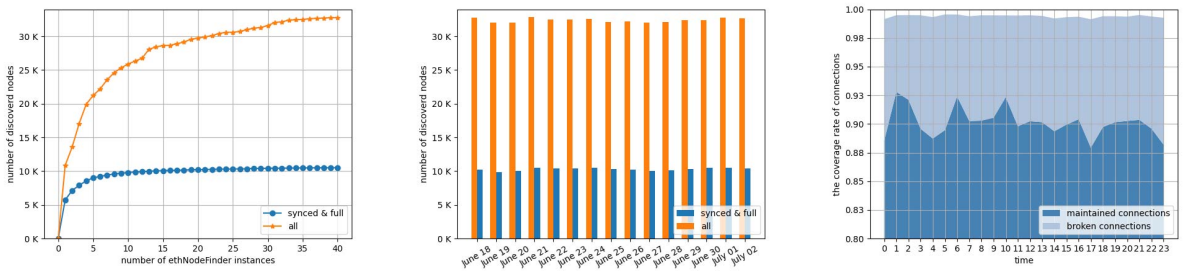
To validate the discovery coverage of ethNodeFinder, we compared the result with two external data sources, ethernodes.org [11] and ethereum/discv4-dns-lists [12]. The former is a third-party Ethereum node explorer, and the latter is a periodically updated Ethereum reachable node list maintained by Geth team. We crawled ethernodes.org at midnight on July 1st and obtained 3,692 nodes totally. Filtered by the field 'lastSeen', only 1,431 nodes were collected by ethernodes.org in the 24 hours. The ethereum/discv4-dns-lists updated three times in July 1st. By checking the field 'lastResponse', there are 3,017 nodes that remained. For ethNodeFinder, we select nodes discovered on July 1st and with networkID 1 and genesis hash 0xd4e5...8fa3 for comparison, which amounts to 32,764 nodes. As shown in Table I, the total number of discovered nodes of ethNodeFinder instances is far more than that of external data sources. Nodes collected by ethernodes.org and ethereum/discv4-dns-lists can be discovered by ethNodeFinder with the probability of 97.55% and 94.10% respectively. It indicates that ethNodeFinder has a good ability of discovery.

TABLE I
Ethereum Node Discovery Comparison

Datasets (count)	ethernodes (1,431 / 3,692)	discv4-dns-lists (3,017 / 3,037)
ethNodeFinder (32,764)	97.55% (1,396)	94.10% (2,839)

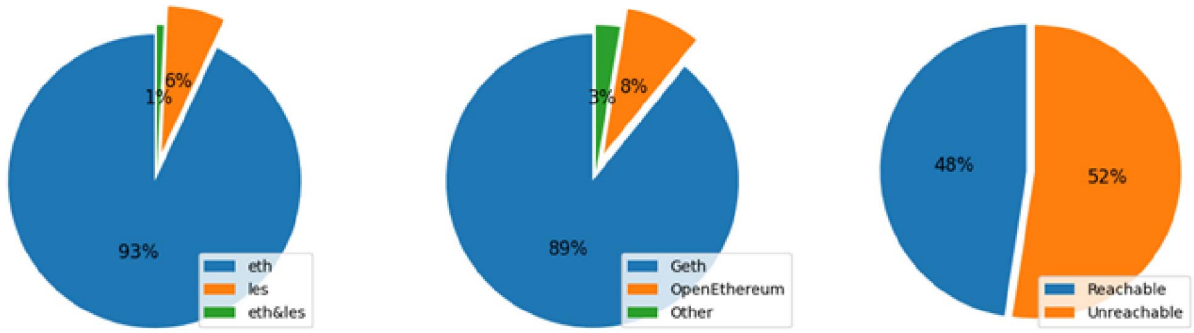
To further verify the coverage, we analyzed the relationship between the number of deployed instances and the total number of discovered nodes. As depicted in Fig 3(a), the total number of all discovered nodes continues to grow and finally reaches over 30 thousand, whose sign of convergence is not obvious. In contrast, the growth rate of completely synced full nodes decreases gradually, and converges to approximately 10 thousand when the number of instances reaches 20. Compared with 20 instances, the number of synced full nodes collected by 40 instances has only increased by 2%. The daily number of discovered nodes is shown in Fig. 4(c). There are around 32K distinct nodes (with networkID 1 and genesis hash 0xd4e5...8fa3) discovered per day on average, among which around 10K are completely synced full nodes. The remaining 22K or more nodes include unsynced nodes, forked nodes, and light nodes, none of them participate in the broadcast of transactions generated by other nodes.

Based on the collected data on July 1st, we analyze the distribution of the nodes. As shown in Fig. 4(a), 6% of Ethereum nodes are light nodes. Light nodes can only create their transactions but not relay transactions of other nodes, therefore these nodes have no anonymity. In terms of client heterogeneity (Fig. 4(b)), we observed that Geth clients account for 89% of all nodes, followed by OpenEthereum at 8%,



(a) (b) (c)

Fig. 3. (a) Total number of discovered nodes for varying numbers of ethNodeFinder instances. (b) Daily number of discovered Ethereum nodes. (c) The connections coverage rate of the supernode.



(a) (b) (c)

Fig. 4. (a) The capabilities of Ethereum nodes. (b) The client implementations of Ethereum synced mainnet full nodes. (c) The network types of Ethereum synced mainnet full nodes.

while the remaining 15 types of clients only account for 3% of all nodes. That's to say, most Ethereum nodes broadcast transactions through *semi-flooding* propagation mechanism. Besides, Fig. 4(c) indicates there are more unreachable nodes than reachable nodes in the Ethereum network. Specifically, unreachable nodes account for 52% of all synced full nodes. It means that the supernode must allow incoming connections to monitor unreachable nodes.

B. The connections coverage

Experiment Setup. The ethTxListener instances on mainnet were deployed in a distributed manner to cope with the huge traffic. Specifically, 10 instances were deployed on 10 Ubuntu 16.04 machines with 8GB RAM and 4cores CPU located all in America. Each instance maintains connections with non-overlapping parts of nodes in the entire network. The Network Time Protocol (NTP) is used for clock synchronization between instances.

In order to obtain the approximate coverage rate of the supernode, we compare the number of nodes currently connected to the ethTxListener instances with the number of nodes discovered by ethNodeFinder instances in the last hour. The comparisons were made at each hour on July 1st. As shown in Fig. 3(c), more than 99% of the nodes had ever established a connection with the supernode in the day, and around 90% were maintaining the connection. For the broken connections, the main reason (over 65%) of broken is the instability of the network.

C. Deanonymization with basic estimators

Experiment Setup. The deanonymization experiments were carried out on the testnet (Ropsten) and mainnet of Ethereum. The setup of ethTxListener instances on mainnet has been described in Section IV-B. For testnet, three independent ethTxListener instances were deployed respectively in America, Germany and India. Each instance ran on Ubuntu 16.04 machines with 8GB RAM and 2cores CPU.

To verify the success rate of the deanonymization attacks, we built **ethTxPretender** to generate transactions and sends them only to one selected node (i.e., the target node) each time. These transactions from ethTxPredenter will behave just like that from the target node. For deanonymization experiments on testnet, 300 nodes were randomly chosen by ethTxPretender as the target nodes. For each target node, the ethTxPretender created 10 transactions and sent them only to the node, amounting to 3000 transactions. For deanonymization experiments on mainnet, since real money is required to spend the transactions fee, we randomly selected 100 nodes as the target nodes, and only one transaction was sent to each node, a total of 100 transactions.

The result of deanonymization on testnet is shown in Table II. For FirstReach estimator, the success rates of ethTxListener instances in India, Germany, and America are 77.80%, 64.80%, and 76.77% respectively. Compared with FirstReach estimator, the performance of FirstSent estimator is better on the success rates (82.43%, 85.30%, and 83.97% respectively), which is less affected by geographical location.

TABLE II
THE SUCCESS RATES OF DEANONYMIZATION ON ETHEREUM TESTNET.

Estimator \ Success Rate	IN	DE	US
FirstReach	77.80%	64.80%	76.77%
FirstSent	82.43%	85.30%	83.97%

The ultimate goal of deanonymization is to identify the source node of a transaction directly, but reducing the size of the anonymous set is also meaningful. In Fig. 5(a) we depict the success rates that the source node is in the set of top k nodes the estimator outputs. The performance of FirstSent estimator is throughout better than the FirstReach estimator. When $k = 10$, the success rates of FirstSent estimator for ethTxListener instances at different geographical locations all reach 98%. That's to say, FirstSent estimator is able to reduce the anonymous set of a transaction to 10 nodes (about 1% of all nodes on testnet) with 98% success rate.

Recall that transactions in Ethereum can only be generated from the owner of the EOA address (i.e., the sender address), we can simply believe that transactions with the same sender address should be sent from the same node. Based on this assumption, the sender's address can be deanonymized through multiple transactions. As shown in Fig. 5(b), with the help of three transactions, FirstSent estimator located in Germany can infer the source node of the sender with a success rate of 99%.

The result of deanonymization on mainnet is shown in Fig. 5(c). When $k = 1$, i.e., the first node that the estimator outputs is the right source node, the success rate of FirstSent estimator is 82%, which is 2% higher than that of FirstReach estimator. When $k = 10$, the success rates of the two estimators are improved to 93% and 91% respectively. The result indicates that FirstSent estimator is able to reduce the anonymous set of a transaction to 10 nodes (about 0.1% of all nodes on mainnet) with a success rate of 93%.

D. Deanonymization with ML-based estimator

By reviewing the experiment results, we find that when FirstSent estimator fails, FirstReach estimator may infer the source node correctly in some cases. Inspired by this, we expect to train a machine learning based estimator to combine the two estimators, so as to infer the source node with a higher success rate without human decision.

Features Selection. The features selected for the classification task of deanonymization are detailed as follows.

1) *Reach_Time_Diff.* The time difference between the arrival timestamp of the node and the minimum arrival timestamp.

2) *Sent_Time_Diff.* The time difference between the estimated sending timestamp of the node and the minimum estimated sending timestamp. The delays were measured based on TCP timestamp.

3) *Inst_Delay.* The delay measured at the closest time (within 1 second) the transaction arrives.

4) *Avg_Delay.* The average delay of all delays measured within 10 seconds before and after the transaction arrives.

5) *Delay_Std.* The standard deviation of all delays measured within 10 seconds before and after the transaction arrives.

Classifiers Selection. Although there are some studies on how to choose classifiers, the most commonly used approach is to test the performance of different classifiers on the current dataset. The datasets obtained in Section IV-C are split into two halves for train and test. The evaluated classifiers include Adaboost, GradientBoosting and RandomForest. The experiment results are shown in Table III. According to the F1 scores, the RandomForest classifier (trained with 50 DecisionTree classifiers) achieves the best performance on all datasets. Therefore, we choose the RandomForest classifier for the following experiments of deanonymization.

TABLE III
PERFORMANCE OF BINARY CLASSIFIERS.

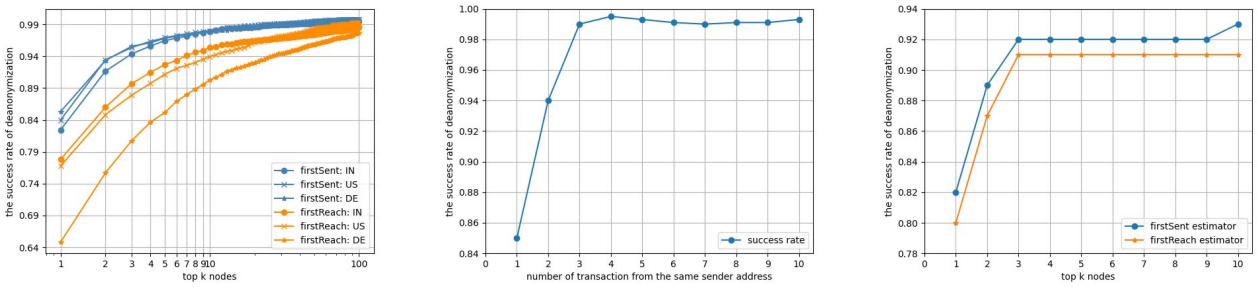
	Algorithm	Precision	Recall	F1
mainnet	Adaboost	67%	84%	75%
	GradientBoosting	85%	87%	86%
	RandomForest	85%	95%	**90%**
testnet-DE	Adaboost	84%	84%	84%
	GradientBoosting	89%	86%	88%
	RandomForest	88%	89%	**88%**
testnet-US	Adaboost	83%	81%	82%
	GradientBoosting	85%	89%	87%
	RandomForest	85%	89%	**87%**
testnet-IN	Adaboost	79%	79%	79%
	GradientBoosting	85%	88%	86%
	RandomForest	83%	89%	**86%**

Evaluation. Finally, we apply the trained binary classifier to infer the source node. As shown in Fig. 6, the ML-based estimator has the best success rates for both deanonymization in Ethereum testnet (88.6%, 89% and 86.3% respectively) and mainnet (88%). Although the performance of the ML-based estimator depends to some extent on FirstReach estimator and FirstSend estimator, it can infer the source node with a higher success rate without human decision.

V. DISCUSSION

Our experiments show that the anonymity of Ethereum P2P network is pretty limited. Ethereum doesn't consider much anonymity when being designed and implemented. Now we will analyze the efficiency and anonymity of its propagation mechanisms.

As mentioned before, there are three different transaction propagation mechanisms in Ethereum, namely *Naive-Flooding*, *Semi-Flooding* and *Gossip*. We first qualitatively analyze the anonymity of these three propagation mechanisms.



(a) (b) (c)

Fig. 5. (a) The top-k deanonymization success rates on Ethereum testnet. (b) Deanonymization of sender addresses with multiple transactions. (c) The top-k deanonymization success rates on Ethereum mainnet.

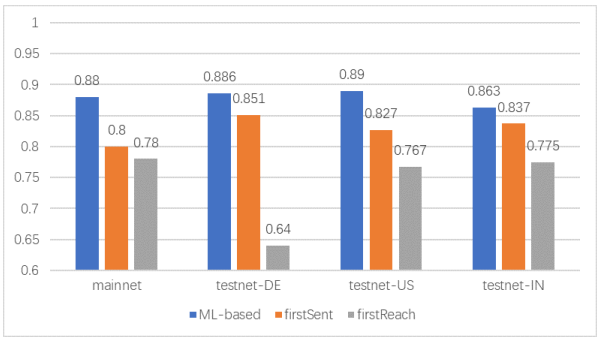


Fig. 6. The comparison of the success rates between ML-based estimator and the two basic estimators.

When the supernode establishes a connection with a Geth node, it can receive all transactions sent by it directly. However, when connecting to an OpenEthereum node, the attacker may not be chosen as the node to send to. Suppose that the supernode maintains m connections with an OpenEthereum node ($m \geq 0$). The probability p that the supernode is chosen as the node to send transactions to can be calculated as:

$$p(n, m) = 1 - \frac{c(n - m, \sqrt{n})}{c(n, \sqrt{n})}. \quad (3)$$

By default, the maximum number of connections of OpenEthereum client is 50. Fig.7(a) depicts the relationship between m and p when n equals 25 and 50. It can be seen that if $n = 50$, when $m = 1$, p equals 14.14%. When m increases to 10, p increases to 81.68%. For *Naive-Flooding* and *Semi-Flooding*, p is always equal to 100% when one connection is maintained. It's worth noting that the anonymity of *Semi-Flooding* is even worse than that of *Naive-Flooding*, since in *Semi-Flooding* the neighbors that only received the transaction hash need more time to request the complete transaction before sending it to their neighbors. Thus when considering anonymity, *Gossip*>*Naive-Flooding*>*Semi-Flooding*.

Then we discuss the broadcast efficiency of three propagation mechanisms. Suppose there are total N nodes in Ethereum network, and the degree of each node is n (i.e., each node maintains n connections with other nodes, so-called neighbors). NetworkX, a Python language library for exploration and analysis of networks and network algorithms, is used to generate a small-world graph close to the current Ethereum network (N=10,000 and n=50). Based on the graph, we simulated the propagation of the three propagation mechanisms. The broadcast efficiency of the mechanisms is shown in Fig. 7(b).In just 4 propagation rounds, a transaction can be known by the entire network with the *Naive-Flooding* mechanism. Although the efficiency of *Semi-Flooding* is slightly better than that of *Gossip*, it both takes 7 rounds for the two mechanisms to broadcast the transaction to the entire network. Thus when considering efficiency, *Naive-Flooding*>*Semi-Flooding*>*Gossip*.

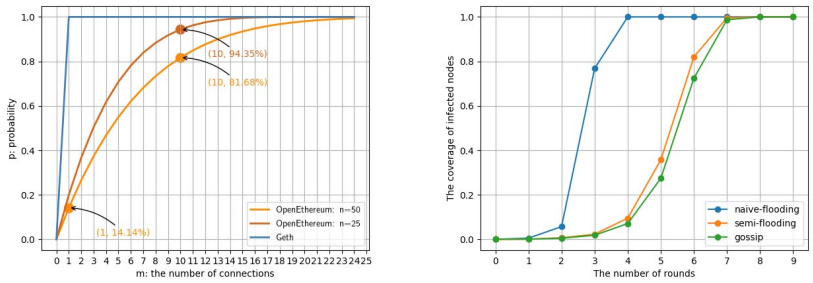
In summary, the broadcast efficiency of *Gossip* is almost the same as that of*Semi-Flooding*, and its anonymity is the best of the three propagation mechanisms. Considering both the efficiency and anonymity, we believe that *Gossip* is a better choice for the Ethereum P2P network.

Ethical Concerns. To validate the feasibility of deanonymization in Ethereum, we conducted our experiments both on testnet and mainnet in a responsible manner. The transactions used for deanonymization were all generated by ourselves. Additionally, we securely delete all collected data after statistically analyzing them, only publish aggregated statistics about the collected data.

VI. Related Work

The existing research on the anonymity and privacy of Ethereum is based on heuristics or extracted features, which aims at the clustering of addresses. In 2020, Victor et al. proposed several heuristics that exploit patterns related to deposit addresses, multiple participation in airdrops and token authorization mechanisms [5]. Besides, Ferenc et al. utilized the active time of the day, the gas price selection and the location in the transaction graph as the features to cluster addresses. They also proposed deanonymization techniques against several popular privacy-enhancing tools [6].

The goal of address clustering is to find multiple Ethereum addresses of the same user, but it can't reveal the user's



(a) (b)

Fig. 7. (a) The probability of being chosen as the node to send transactions to. (b) The broadcast efficiency of different propagation mechanisms.

identity in the real world. Mapping account addresses to their IP addresses provides an opportunity to deanonymize users.

In 2014, Koshy et al. analyzed the propagation of transactions in the Bitcoin P2P network and found that abnormal transaction patterns can be utilized to distinguish the initiator of the transaction from other nodes [7]. For nodes behind NATs or firewalls, Biryukov et al. proposed a deanonymization method that identifies the client of the transaction initiator by its eight entry nodes [8]. In 2015, the Bitcoin community responded to these attacks by changing the network's flooding mechanism from trickle to diffusion. Fanti et al. analyzed the anonymity properties of both mechanisms and concluded that Bitcoin's networking protocols offer poor anonymity properties [9]. In 2019, Biryukov et al. introduced a novel technique for linking Bitcoin transactions based on transaction propagation times analysis [10].

The previous deanonymization attacks based on the analysis of the P2P network mostly focus on Bitcoin. As far as we know, now there is no related research on Ethereum.

VII. Conclusion

In this paper, for the first time we explored the feasibility of deanonymizing Ethereum users based on P2P network analysis. We observed that the attacker can make connections with approximately 90% Ethereum synced full nodes. Based on the well-connected supernode, the deanonymization with basic estimators preliminarily indicate that the anonymity of the Ethereum P2P network is pretty limited. To reduce the influence of network delay, we implemented and evaluated a ML-based estimator, which can further improve the success rate of deanonymization to 88%. By the discussion of the propagation mechanisms, we believe that *Gossip* is a better choice for the Ethereum P2P network.

Acknowledgment

This work was supported by the Key Research and Development Program for Guangdong Province under Grant 2019B010137003, the Beijing Natural Science Foundation under Grant M21037, and the Fundamental Research Program of National Defence (JCKY2019211B001).

References

[1] "GitHub - ethereum/devp2p: Ethereum peer-to-peer networking specifications", April 2021. [Online]. https://github.com/ethereum/devp2p.
[2] Kaminsky, Dan. "Black ops of TCP/IP." Black Hat USA 44 (2011).
[3] Lumezanu, Cristian, et al. "Triangle inequality variations in the internet." Proceedings of the 9th ACM SIGCOMM conference on Internet measurement. 2009.
[4] Wang, Guohui, Bo Zhang, and TS Eugene Ng. "Towards network triangle inequality violation aware distributed systems." Proceedings of the 7th ACM SIGCOMM conference on Internet measurement. 2007.
[5] Victor, Friedhelm. "Address clustering heuristics for Ethereum." International Conference on Financial Cryptography and Data Security. Springer, Cham, 2020.
[6] Béres, Ferenc, et al. "Blockchain is watching you: Profiling and deanonymizing ethereum users." arXiv preprint arXiv:2005.14051 (2020).
[7] Koshy, Philip, Diana Koshy, and Patrick McDaniel. "An analysis of anonymity in bitcoin using p2p network traffic." International Conference on Financial Cryptography and Data Security. Springer, Berlin, Heidelberg, 2014.
[8] Biryukov, Alex, Dmitry Khovratovich, and Ivan Pustogarov. "Deanonymisation of clients in Bitcoin P2P network." Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. 2014.
[9] Fanti, Giulia, and Pramod Viswanath. "Deanonymization in the bitcoin P2P network." Proceedings of the 31st International Conference on Neural Information Processing Systems. 2017.
[10] Biryukov, Alex, and Sergei Tikhomirov. "Deanonymization and linkability of cryptocurrency transactions based on network analysis." 2019 IEEE European Symposium on Security and Privacy. IEEE, 2019.
[11] "The Ethereum Network & Node Explorer", April 2021. [Online]. https://www.ethernodes.org/.
[12] "Ethereum discv4-dns-lists", April 2021. [Online]. https://github.com/ethereum/discv4-dns-lists.
[13] Bonneau, Joseph, et al. "Mixcoin: Anonymity for bitcoin with accountable mixes." International Conference on Financial Cryptography and Data Security. Springer, Berlin, Heidelberg, 2014.
[14] Maxwell, Gregory. "CoinJoin: Bitcoin privacy for the real world." Post on Bitcoin forum. 2013.
[15] Ruffing, Tim, Pedro Moreno-Sanchez, and Aniket Kate. "Coinshuffle: Practical decentralized coin mixing for bitcoin." European Symposium on Research in Computer Security. Springer, Cham, 2014.
[16] Meiklejohn, Sarah, and Rebekah Mercer. "Möbius: Trustless tumbling for transaction privacy." (2018): 881-881.
[17] Seres, István András, et al. "Mixeth: efficient, trustless coin mixing service for ethereum." International Conference on Blockchain Economics, Security and Protocols (Tokenomics 2019). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.
[18] Androulaki, Elli, et al. "Evaluating user privacy in bitcoin." International conference on financial cryptography and data security. Springer, Berlin, Heidelberg, 2013.
[19] Reid, Fergal, and Martin Harrigan. "An analysis of anonymity in the bitcoin system." Security and privacy in social networks. Springer, New York, NY, 2013. 197-223.