Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2023.030468

ARTICLE

"Half of the Node Records Are Forged?": The Problem of Node Records Forgery in Ethereum Network

Yang Liu[1,2,*], Zhiyuan Lin[1], Yuxi Zhang[1], Lin Jiang[1,*] and Xuan Wang[1,3]

[1]School of Computer Science and Technology, Harbin Institute of Technology (Shenzhen), Shenzhen, 518055, China

[2]Peng Cheng Laboratory, Shenzhen, 518055, China

[3]Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies, Shenzhen, 518055, China

*Corresponding Authors: Yang Liu. Email: liu.yang@hit.edu.cn; Lin Jiang. Email: zoeljiang@hit.edu.cn

Received: 07 April 2023 Accepted: 05 July 2023 Published: 17 November 2023

ABSTRACT

Ethereum, currently the most widely utilized smart contracts platform, anchors the security of myriad smart contracts upon its own robustness. Its foundational peer-to-peer network facilitates a dependable node connection mechanism, whereas an efficient data-sharing protocol constitutes as the bedrock of Blockchain network security. In this paper, we propose NodeHunter, an Ethereum network detector implemented through the application of simulation technology, which is capable of aggregating all node records within the network and the interconnectedness between them. Utilizing this connection information, NodeHunter can procure more comprehensive insights for network status analysis compared to preceding detection methodologies. Throughout a three-month period of unbroken surveillance of the Ethereum network, we obtained an excess of two million node records along with over one hundred million node acquaintances. Analysis of the gathered data revealed that an alarming 49% or more of these node records were maliciously forged.

KEYWORDS

Blockchain; ethereum; peer-to-peer networks; node discovery protocol; malicious behavior

1 Introduction

Blockchain is an immutable chain-like structure implemented on a wide area network via asymmetric encryption algorithms and distributed consensus algorithms. Bitcoin and Ethereum, as typical exemplars of blockchain systems, are both constructed on an underlying peer-to-peer network.

As the application of Ethereum broadens, research into the security of Ethereum's application layer has consequently been intensified, encompassing aspects such as smart contract malicious code detection, consensus algorithm security analysis [1], and monitoring of transaction trends [2]. Nonetheless, existing research and analysis focused on the foundational peer-to-peer network of Blockchain are inadequate, with a significant majority concentrated within the realm of Bitcoin [3]. Furthermore, attributed to the open nature of the Ethereum protocol, its network protocol complexity surpasses that of Bitcoin [4]. In other words, the insights derived from Bitcoin research cannot be directly and seamlessly applied to Ethereum. Given these circumstances, there exists a necessity for a

comprehensive detection and analysis of the Ethereum peer-to-peer network, to identify potential risks and hidden threats. In response, we devised a novel methodology for the detection of the Ethereum peer-to-peer network and developed it into an open-source tool, NodeHunter. Through a quantitative analysis of the obtained data, we discern the current operational state of the Ethereum network along with the existing security threats within it. Finally, we analyze and interpret the observed phenomena.

In this paper, we propose a novel detection mechanism, which utilizes the feature of the FindNode packet in discv4 protocol to access all records stored in a node's local node database within the network. We refer to these records as the 'acquaintance node' of the respective node.

During our analysis of the experimental data, we discovered that the entire peer-to-peer network was significantly more complex and chaotic than anticipated. And we also discovered a widespread problem of node records forgery in peer-to-peer networks. For clarity in analysis and description, we refer to the normally operating node as an 'active node,' and those disseminating forged node records are termed 'malicious nodes'. A malicious forgery of node records involves an attacker generating a large number of random public keys, which malicious nodes then encapsulate into node records and disseminate broadly. A node record refers to the information related to an Ethereum node and the interconnectedness between nodes, as captured by the detector. Nodes engaged in forgery may attempt to double-spend coins, generate invalid blocks, tamper with consensus mechanisms, or cause network disruption. They could also collaborate with other malicious nodes or attackers to instigate more sophisticated assaults. For instance, a '51% attack' could occur when a faction of these forging nodes seizes control of more than half of the network's computational power, thereby gaining the ability to arbitrarily rewrite blockchain history. Given these potential threats, it is crucial to identify and restrain such forgery nodes from infiltrating or manipulating the network.

Contrasted with the earlier method [5], the detection technique presented in this paper incorporates a recursive search process for each node detected. This step leverages the data exchange features inherent in the node discovery protocol, enabling us to access data held in every node's own node database. By analyzing this data, our detection method can more comprehensively acquire the node record information present throughout the entire Blockchain network, subsequently uncovering instances of node record forgery. In summary, this paper offers two main contributions.

First, we developed NodeHunter, an open-source tool capable of gathering data on the acquaintance relationships between nodes that has never been accessed before. NodeHunter can continuously scan and monitor Ethereum's peer-to-peer network, efficiently generating network snapshots. Additionally, we have enhanced the detection methodology by leveraging the node database used by mainstream Ethereum clients, enabling us to acquire and analyze previously unexamined node record data.

Second, we examined the overarching state of the Ethereum network and identified a malicious node records forgery issue. Further, we analyzed the causes, impacts, and countermeasures associated with this issue. Based on the most conservative estimates, 49% of all node records were randomly generated invalid records, posing a risk to the efficiency of node discovery and compromising the security of the Blockchain network.

The following part of this paper is organized as follows: Sections 3 and 4 present the operational principle of the detector. Section 5 delves into the analysis of the collected data. Finally, Section 6 highlights the phenomenon, causes, effects, and countermeasures pertaining to node records forgery.

2 Background

Currently, Ethereum holds a market value surpassed only by Bitcoin [6], and it stands as the most extensively utilized platform for smart contracts [7]. Despite the growing body of research focused on security analysis of Ethereum's consensus algorithm and contract code, attention to the peer-to-peer network characteristics of Ethereum remains relatively lacking. Research indicates that the security of blockchain systems employing Proof of Work mechanisms hinges on the reliability of their peer-to-peer networks [8]. In addition, existing studies on Bitcoin shows that driven by vested interests, some large mining pools may launch DDoS attacks on their counterparts. The probability of sending DDoS attacks closely correlates with local computing power and attack cost. Moreover, it is observed that a mere 2% of nodes, advantaged by favorable network conditions, command 75% of Bitcoin's computing power, making the Bitcoin network susceptible to both DDoS and Sybil attacks [9]. Evidently, the network layer of blockchain systems faces a multitude of challenges. In parallel, several scholars have suggested various traffic detection methods to counter these attacks [10–15].

Ethereum's peer-to-peer network predominantly operates two protocols-the Node Discovery Protocol, which is based on UDP, and the RLPx Protocol, which relies on TCP [4]. The Node Discovery Protocol facilitates querying of the nearest node records among nodes [16]. Upon selecting the target node, the local node initiates an encrypted data communication link with it through the RLPx protocol [17]. Fig. 1 illustrates the two protocols encompassed within the Ethereum peer-to-peer network and the primary steps entailed in each.

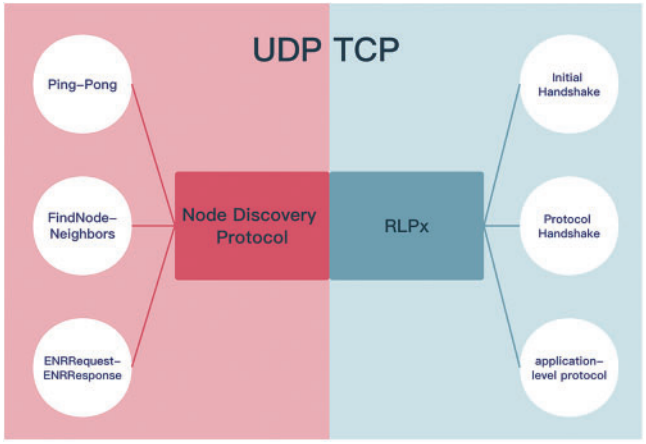

Figure 1: Overview of Ethereum network protocol

2.1 Node Discovery Protocol

Within the Node Discovery Protocol, nodes engage in gossiping to identify other nodes on the network. To partake in the Ethereum network and execute the block synchronization process, the detector utilizes a set of bootstrap nodes. These nodes discover other active nodes and store their data in the bucket [5].

To enable swift communication of node records between nodes, Ethereum's Node Discovery Protocol employs UDP as the transport layer protocol. The v4 version of this protocol encompasses six message types: Ping, Pong, FindNode, Neighbors, ENRRequest, ENRResponse, each of these serves as a responsive packet to one another. The Ping and Pong packets ascertain the status of the remote

node, while the Neighbors packet returns 16 records each time, representing the nodes closest to those in the FindNode packet [16].

The Node Discovery Protocol forms a P2P overlay network by incorporating Ethereum encoding and encryption into the Kademlia algorithm. In this arrangement, the node ID primarily utilizes the hexadecimal public key, and the node distance is determined as the XOR logical distance between each pair of node IDs. All Ethereum clients, through the RLPx protocol, amass peer information for each logical distance in a bucket, which has a maximum size of 16. In the current version's implementation, the default packet timeout is set at 500 ms [18].

Fig. 2 illustrates the execution process of the Node Discovery Protocol and the addition of node records to the node database. During the first Round-Trip Time (RTT), the local node dispatches a FindNode packet while receiving the Neighbors packet returned by the remote node, storing the 16 node records contained within the Neighbors packet to the node database [19]. During the second RTT, the local node sends a Ping packet to the remote node to attempt activation. If an activation packet is received during this interval, it signifies that the remote node is in an active state. This process then repeats cyclically for the remaining nodes.

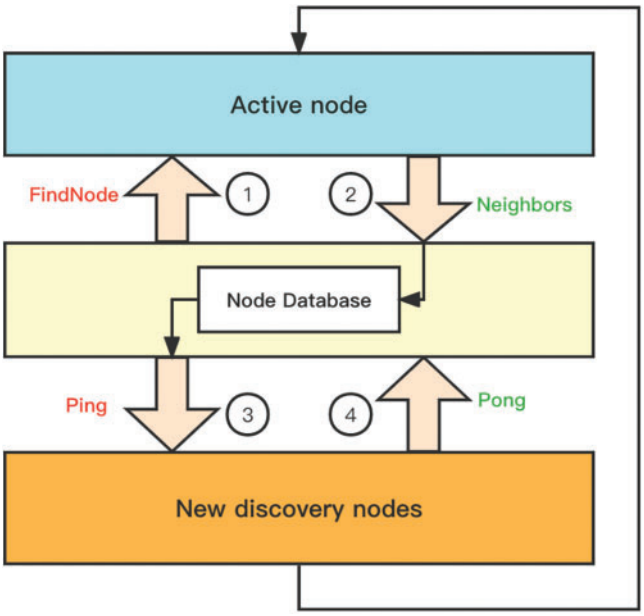

Figure 2: The process of inserting data into node database

2.2 Node Record and Distance between Nodes

Every peer within the Ethereum network possesses its own unique node records, identified by their respective node IDs. A node ID is a 512-bit (64-byte) cryptographic ECDSA public key, and is represented as a hexadecimal number within the node record.

The measure of distance between two nodes is referred to as 'logdist.' Ethereum employs a modified version of the Kademlia protocol for its logdist function, wherein all node records known to the local node are arranged in order of logdist.

The process to calculate logdist comprises two steps. Firstly, the node IDs are hashed using the keccak256 algorithm, resulting in 256-bit values. Subsequently, the XOR result of the two hash values is calculated, and the logarithm of this XOR result is taken.

2.3 The RLPx Transport Protocol

The step following node discovery involves data exchange with the discovered nodes. This process is facilitated by the RLPx protocol, which serves to establish a secure TCP connection with a remote peer [17]. The process of connection establishment involves two handshake steps: an encryption handshake and a subprotocol handshake.

Encrypted handshake: This step involves constructing the symmetric encryption key based on the Elliptical Curve Integrated Encryption System (ECIES), which will be used for subsequent communication.

Subprotocol handshake: In this step, the communicating parties exchange the names and versions of the subprotocol, selecting the most suitable protocol for the transmission of application layer data. Every node initiates the communication by sending a HELLO message to its peer. This message includes crucial information such as the node's ID, the DEVp2p version, client name, supported application protocols/versions, and the listening port number, which defaults to 30303.

3 Principle of Detector

This section elucidates the acquaintance relationship among nodes and outlines the data acquisition process implemented by the detector.

3.1 Node Acquaintance Relationship

We define the node records stored in the local database as this node's acquaintance nodes. The node discovery protocol specification dictates that each returned Neighbors packet will include 16 node records, which are closest to the ones dispatched by FindNode in the database. Previous detection methodologies overlooked the node records stored in the node database and required only a single round trip of FindNode and Neighbors packets for each node's detection, resulting in a relatively restricted number of node record entries.

In this paper, we employ a method of repeated querying and aggregation-based de-duplication to retrieve all node records from the databases of remote peers. We refer to the asymmetric relationship in which one node stores the node records of another node as the node acquaintance relationship.

3.2 Data Acquisition Method

The comprehensive process of a simulation node operating within the network, as depicted in Fig. 3, comprises two principal steps: bootstrap node initialization and recursive search.

Initially, the simulation node procures a set of seed nodes from the default bootstrap node. Subsequently, it initiates a recursive search on each seed node until no new nodes emerge, signifying the completion of the Ethereum network exploration.

The query process for each node bifurcates into two stages. The first stage involves acquiring all known nodes from each node via the node discovery protocol. The second stage entails establishing a TCP connection with the node and obtaining its corresponding metadata by simulating the handshake process of the RLPx protocol. Through these procedures, we can amass the final detection dataset.

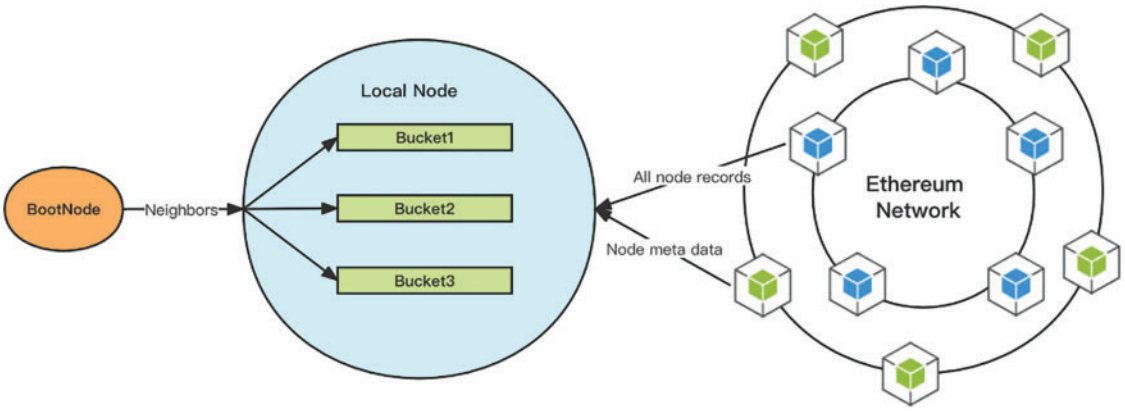

Figure 3: Process of collecting data from Ethereum network

Every node within the Ethereum network possesses its unique node record, identifiable by the node ID. The node ID is a 512-bit ECDSA public key represented in hexadecimal format. These node records are data structures encapsulating information regarding the node's identity, location, and capabilities in the Ethereum network. Nodes independently generate these records using cryptographic keys and store them in a levelDB database, facilitating efficient lookup and routing. Through a gossip protocol, nodes disseminate their records to other nodes, bolstering network connectivity and resilience. To reflect changes in status and availability, nodes periodically update and maintain these records.

4 Overall Analysis of Data

This section presents the experimental setup and provides an analysis of the collected data based on predefined parameters to ascertain the prevailing operational status of the Ethereum network.

4.1 Experimental Configuration

The detector, compiled in the Go language, operates under version 1.17. To enhance data access efficiency, all node records and acquaintance relationships are preserved in LevelDB, managed by simulating a table structure through prefix coding. The detector also provides a remote procedure call (RPC) interface for real-time monitoring of data collection progress and the prevailing network status.

Between April 2nd, 2022 and June 15th, 2022, the Ethereum network was monitored for a quarter, resulting in the collection of more than 33 GB and over 200 million units of raw data. The detector was deployed on a server equipped with an Intel(R) Xeon(R) Gold 5220 CPU @ 2.20 GHz *72 CPU, 128 GB RAM, and a 2 TB SSD, running on the Linux 18.04 operating system. The data analysis program, designed in Python 3.9, employed LevelDB as the local database for data classification and storage.

4.2 Node Meta Data

Node metadata encompasses the node record serial number, client version, development language, operating system, and supported application layer protocols. The node record serial number is termed as SEQ, and its value can be acquired from the node discovery protocol. Due to disparities in the implementation of Ethereum node records standards between Geth and Parity clients, the SEQ of the Geth client increments from zero, whereas the Parity client opts to use a random number as the

initial value. Hence, the SEQ value of the Geth client can serve as a measure of node activity. Upon completion of the encryption handshake process in the RLPx protocol, the protocol handshake needs to be carried out further. In this stage, both parties are required to exchange the client's name and supported sub-protocols, enabling the acquisition of the remaining metadata.

Examples of data that can be gathered during the protocol handshake include:

1. Geth/v1.10.13-stable/linux-amd64/go1.17.5 les/2, les/3, les/4
2. go-opera/v1.0.2-rc.5-3002f17a-1630337195/linux-amd64/go1.16 opera/62

In summary, the procedure for obtaining node metadata can be segmented into two steps:

1. Acquisition of node record serial number: the simulation node mimics the sending of an ENR-Request Packet to the remote node and deciphers the SEQ from the received ENRResponse Packet.
2. Gathering of remaining metadata: the simulation node and the remote node initially complete the ECIES handshake of the RLPx protocol, and then the two entities will conduct the protocol handshake. At this stage, we can extract the client version, application layer protocol, and other details from the other party's handshake packet. Based on the Ethereum client's source code analysis, a remote node returning the error message "too many peers" is deemed to be a highly active node with over 50 peers.

4.3 Node Definition

In this paper, we categorize nodes based on their distinct characteristics and behaviors, encompassing active nodes, core nodes, routing nodes, acquaintance nodes, and malicious nodes.

Active node: A node is classified as an active node if it satisfies any of the following conditions:

1. The node returns non-error metadata during the handshake phase of the RLPx protocol.
2. Non-empty ENR records are obtained via the node discovery protocol v5.
3. The node establishes a connection with other nodes at least once (connects to more than one node).

Active nodes denote the standard nodes within the Ethereum network and serve as the bedrock for all peer-to-peer networks. Exploring these can yield insights into the Ethereum network topology and assist in understanding the safety and stability of Ethereum's operations.

Core node: A node is classified as a core node if it fulfills all the following conditions:

1. The node operates the latest client version, suggesting active maintenance.
2. The node maintains a peer count exceeding 50, indicative of high activity levels.
3. The acquaintance relationship, when fed into the PageRank algorithm, results in a weight greater than 0.15‰.

Core nodes are diligently maintained and command a central position within the blockchain network. From a block production standpoint, these nodes harness the majority of the computational power, categorize as high-value nodes, and are most susceptible to various types of attacks.

Routing node: Among the active nodes, certain nodes, denoted as routing nodes, do not execute any application-layer protocol; instead, they merely forward node records. These routing nodes play a vital role in enabling new nodes to swiftly join the blockchain network.

Acquaintance node: An acquaintance node is characterized as a node that has been discovered by the local node through the node discovery protocol and subsequently stored in the node database.

Malicious node: A malicious node is a unique entity that does not execute any blockchain protocol and maintains no acquaintance relationships with other nodes. Instead, it disseminates an extensive array of fabricated node records or spurious node identities to establish connections with active nodes. Malicious nodes pose a threat to the efficiency of the Ethereum network and can significantly compromise the security of certain nodes.

4.4 Network Overview

This research evaluated the data records gathered over two quarters. The primary analysis was conducted on data from the first quarter, which was then supplemented with data procured in the second quarter. In the first quarter, we obtained a total of 1,805,970 node records and 61,221,945 node acquaintance relationships. In the second quarter, we obtained 2,578,458 node records and 124,972,682 node acquaintance relationships, approximately doubling the count from the previous quarter. After the removal of data anomalies such as duplicates and fraudulent records, the active node count stood at 41,439 in the first quarter and 64,559 in the second quarter. This reflects a growth of over 30,000 active nodes, when compared to the detection methodology employed in March 2020 [20,21]. Within the group of active nodes, 1,516 nodes in the first quarter and 2,039 nodes in the second quarter did not operate any application layer protocol and were thus categorized as routing nodes.

We evaluated the correlation between the two detection data sets and the computational hashrate of the entire network, with the results presented in Table 1. Upon unique identification and removal of active and routing nodes based on their public keys, it was observed that the network size had expanded by approximately 25% during this quarter, aligning with the upward trend of the network's computational hashrate [22].

Table 1: Number of nodes change

Node type	First quarter	Second quarter	Growth rate
Node records	1805970	2578458	42.77%
Active nodes	41439	64559	55.79%
Core nodes	287	383	33.45%
Routing nodes	1516	2039	34.50%
Acquaintance relationship	61221945	124972682	104.13%
Network hashrate	870 GH/s	1040 GH/s	19.54%

- Hashrate data from Etherscan.

4.5 Geographical Distribution

In this experiment, the active node records comprise the public key, IP, and port number. Hence, active nodes can be grouped based on the IP data. Fig. 4 depicts the distribution of active nodes collected from 83 countries. Table 2 displays the top five countries ranked by the number of active nodes within the Ethereum peer-to-peer network, and it is noteworthy that these top 5 countries account for 85.98% of all active nodes. The majority of active nodes were found in the United States (43.48%), followed by Austria (18.77%), and Germany (17.69%).

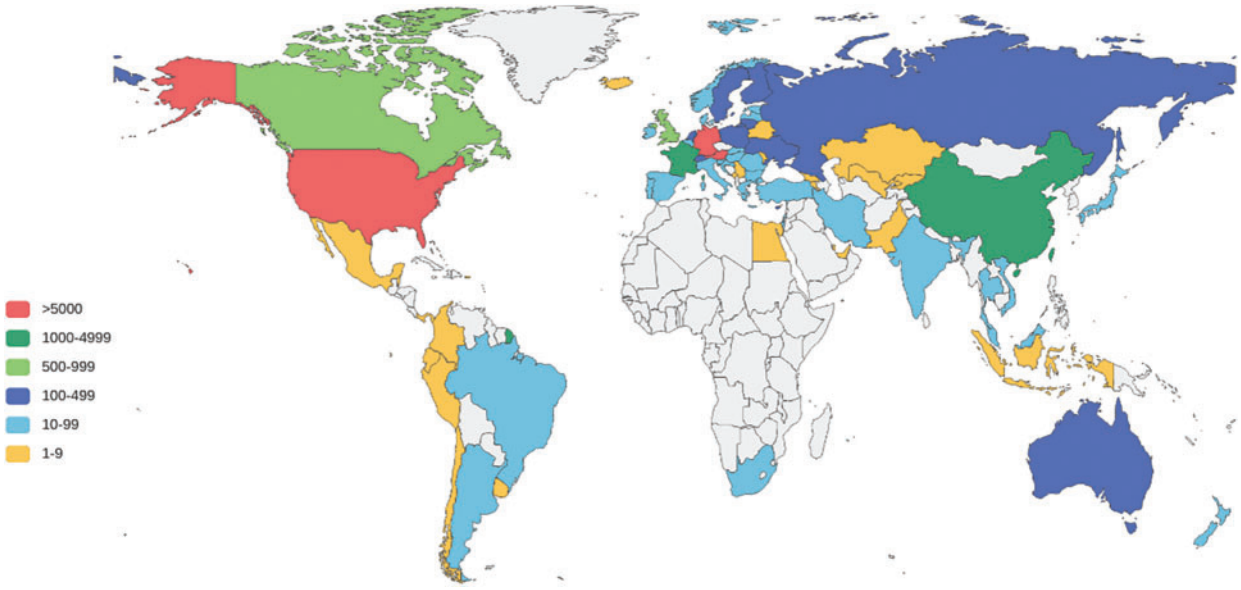

Figure 4: Active node geographic distribution

Table 2: Distribution of active node countries

Country	Number of active nodes	Percentage
United States	28070	43.48%
Austria	12118	18.77%
Germany	11420	17.69%
France	2208	3.42%
China	1691	2.62%

4.6 Metadata Analysis

Through the analysis of metadata garnered from the RLPx handshake protocol utilized by active nodes, we can obtain details pertaining to the node's client version. Fig. 5 presents a statistical distribution of node metadata, detailing the predominant client names, Ethereum application layer protocols, operating system types, and client programming languages. Across the entire peer-to-peer network, there are 82 distinct clients deployed across a variety of nodes, with Geth, OpenEthereum, ParityEthereum, Bor, and Egem constituting the top five. As displayed in Fig. 5, Taifang's Geth client services the most significant portion of nodes, comprising about 54% of the total nodes identified. The Go language emerges as the most popular, employed by a total of 9,420 nodes. Go1.17.2 is the most widely adopted version, utilized by 1004 nodes, accounting for roughly 8% of the total. Additionally, around 8300 Ethereum clients operate using the eth/66 version.

4.7 Core Node and Routing Node

The acquaintance relationship between nodes is non-reflexive. All acquaintance relationships captured by the detector are aggregated into edges, thereby forming the largest connected subgraph within the entire peer-to-peer network. Previous research [23] indicated that the data collected by

the detector is compatible with the PageRank algorithm. Consequently, we utilize the relationships between all active nodes as our input, subsequently calculating the weights of these nodes, and identify 383 nodes that align with our definition of core nodes. Upon comparison, it is evident that the weight of a routing node far exceeds that of a common active node. Fig. 6 displays the proportion of various nodes identified in our analysis. Separating nodes into geographical regions [24] and correlating with node weights, we constructed the global node connection diagram depicted in Fig. 7.

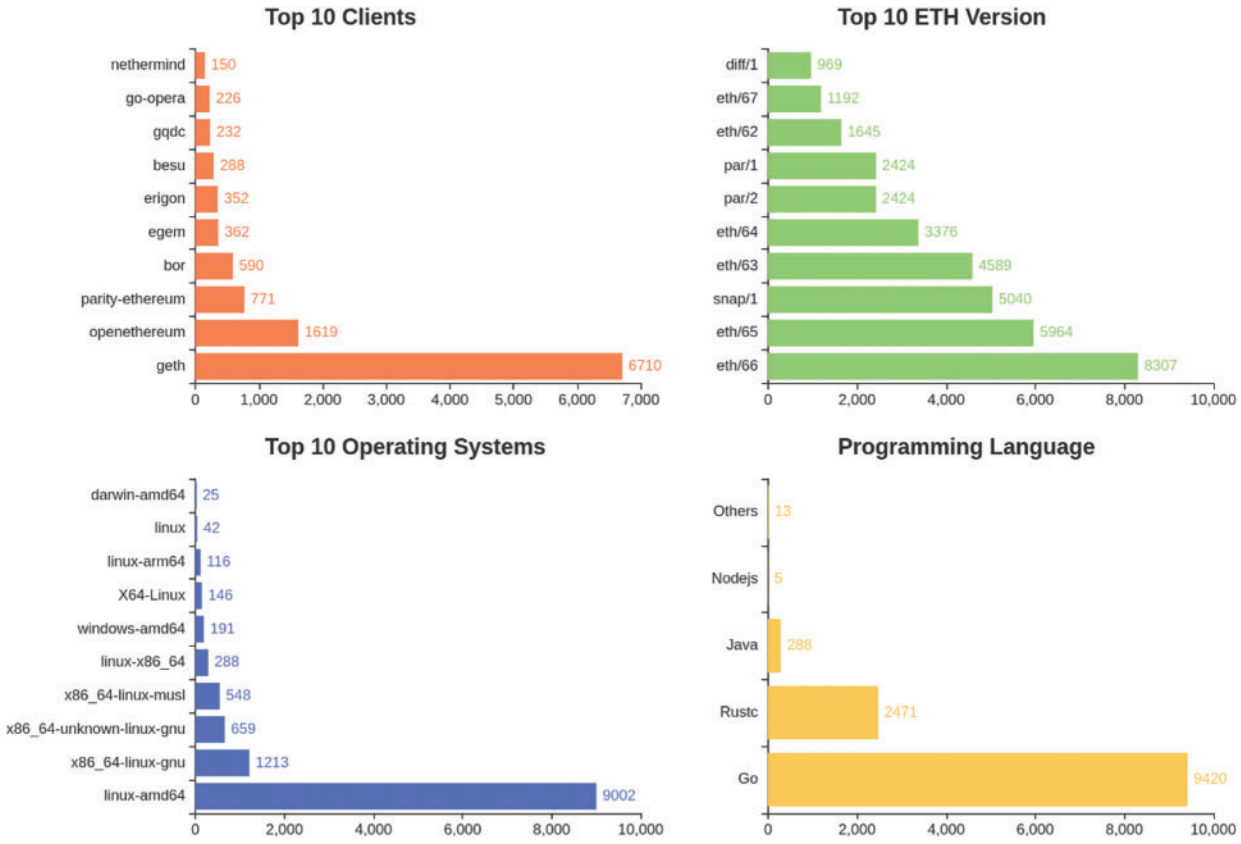

Figure 5: Metadata analysis

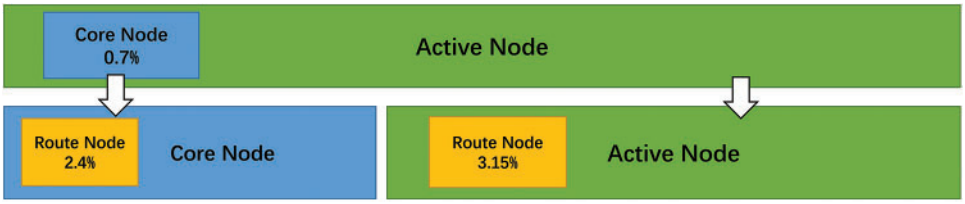

Figure 6: Proportion of various nodes

As shown in Fig. 7, there is a clear geographic aggregation trend within the entire peer-to-peer network. Core and routing nodes establish the backbone of this network, with remaining active nodes predominantly clustering around these core and routing nodes. This pattern not only highlights the distinct roles of core nodes, routing nodes, and regular active nodes within the network, but also underscores the centralized evolution trend of the peer-to-peer network. Experimental results reveal

that the time to complete the detection process nearly doubles upon the removal of routing nodes, thereby emphasizing their pivotal role in enhancing node discovery efficiency.

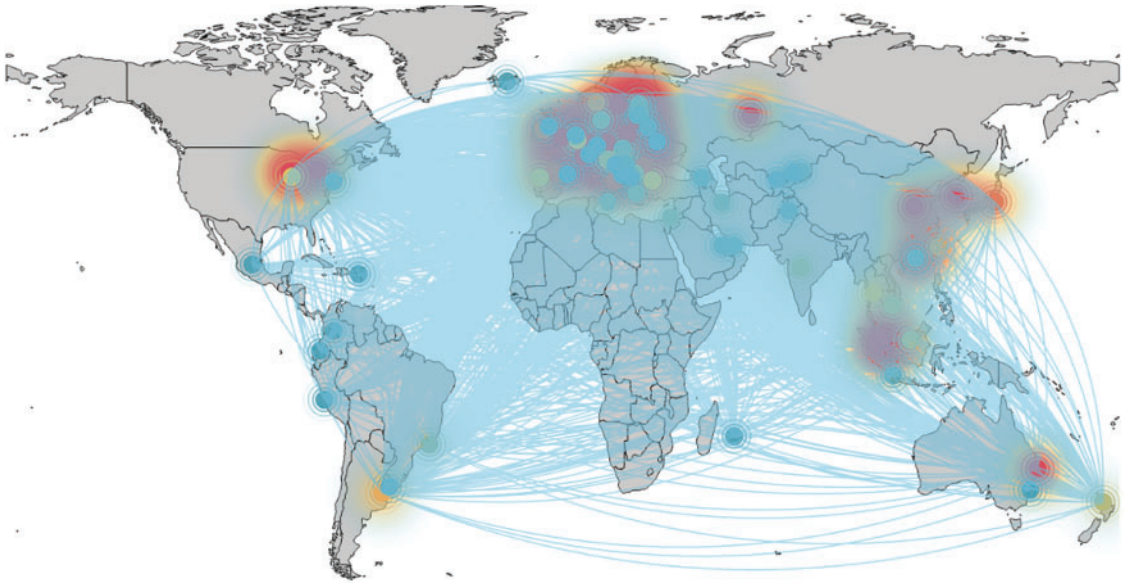

Figure 7: Global node connection diagram

4.8 Experimental Result

During our experimentation, we identified a total of 2,578,458 node records, 124,972,682 acquaintance relationships, and 64,559 active nodes within these raw data sets. Notably, the number of active nodes has doubled compared to the data from two years prior [20], suggesting a rapid expansion in the scale of the Ethereum network in recent years. Furthermore, our method discovered a greater number of active nodes than other current methodologies such as ethernodes and etherscan. This suggests that our novel approach provides a more comprehensive representation of the Ethereum network's nature.

5 Node Records Forgery

This section delves into the principles, objectives, implications, and countermeasures related to the issue of node record forgery.

5.1 Proportion of Forged Records

A node record comprises a triplet: the node ID, node IP, and running port. Records with identical IP addresses but varying node IDs are referred to as duplicate broadcasts. A comparison of these duplicate broadcasts with active nodes reveals that the majority of duplicate broadcasts are invalid records, while a minor portion originates from core nodes. Typically, such duplicate broadcasting occurs only when the client undergoes a reset or when multiple clients share a single public IP. However, in real environments, the incidence of duplicate broadcasting significantly surpasses the standard range. According to the data provided by Ethernodes, the number of nodes from the same IP does not usually exceed one hundred. Furthermore, this phenomenon cannot be attributed to Network Address Translation (NAT) as node records from the same IP generally share the same port (predominantly the default port 30303).

Drawing from the definition of malicious nodes presented in Section 4.3, we align the rules of the node relational database with the node information database to identify forged node information in the current network. Fig. 8 illustrates the proportion of forged node records under different thresholds. To prevent the potential misclassification of normal records, we adopt a conservative threshold of 1000 for analysis in this section. As observable from the figure, almost half of all node records are randomly counterfeited for a variety of reasons.

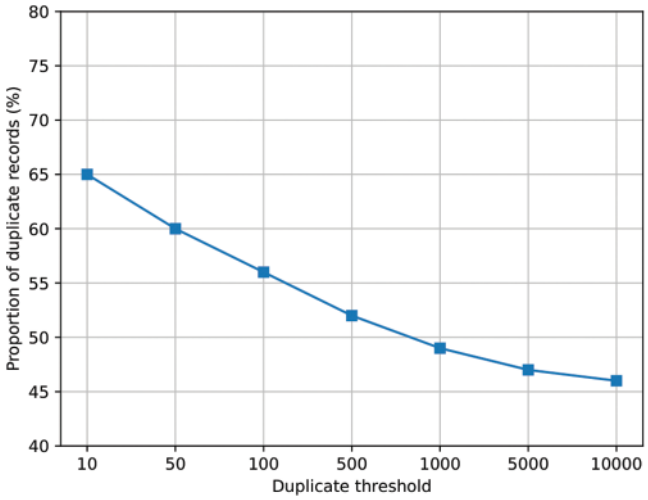

Figure 8: Proportion of forged records

The node ID corresponds to the public key of the node's private key and is denoted by a 128-character long hexadecimal number in the node record. Through analysis of the forged node ID's prefix, we discern that the distribution of the node ID prefix exhibits two distinct characteristics. The first characteristic, manifested as an even distribution across the entire probability space, is termed "random forgery". The second characteristic exhibited through an uneven concentration on certain specific prefixes, is identified as "centralized forgery".

5.2 Random Forgery

A prevalent type of node records forgery involves the random generation of a large number of node records, a tactic we refer to as random forgery. An example of random forgery can be seen when analyzing the node records from the 94.xx.xx.xx address. The examination of a total of 43,643 node records reveals a remarkably uniform distribution of the forged node IDs, indicative of an entirely random generation process.

Forging a node record incurs virtually no cost. An attacker simply needs to generate a 256-bit private key and subsequently execute the ECDSA key generation algorithm. Given that multiple Ethereum nodes can operate from the same network address, and the Ethereum protocol design doesn't impose a limit on the number of node records on the same network address, an infinite number of node records can be generated and broadcast from a single physical machine.

The intent behind random forgery can be dual: The intent behind random forgery can be twofold: firstly, to decrease the operational efficiency of the node discovery protocol, and secondly, to forge more connections, accelerating the confirmation rate for blocks or transactions. Data analysis from our detection indicates that random forgery can indeed impair the operational efficiency of the

node discovery protocol. For instance, by examining the forged node records originating from the 94.xx.xx.xx address, we observed the contamination of the node database of 2777 active nodes. Furthermore, new nodes joining the network are likely to encounter forged records with every 1.2 FindNode packets sent during their initial node detection phase, potentially leading to considerable inefficiencies in network operations.

To assess the effect on the speed of block or transaction confirmations, we constructed a simulation environment comprising 102 nodes with a communication delay of 50 ms. Of these, 100 were ordinary nodes, in addition to nodes A and B for comparative analysis. Node A consistently established connections with ordinary nodes. In contrast, Node B defaulted to establishing connections with ordinary nodes but continued to outwardly broadcast forged node records. All other parameters adhered to the node's default configuration.

To empirically assess the impact on block confirmation, our experiment was structured such that only nodes A and B were granted permission to produce blocks simultaneously across the entire network. The experiment concluded when the count of produced blocks reached a hundred, after which we calculated the quantity of confirmed blocks associated with node B. To determine the impact on transaction confirmation, we designed a scenario where nodes A and B concurrently initiated transactions, while the remaining nodes continued to produce blocks under normal conditions. With a repetition of a hundred transactions, we counted the number of transactions that node B got confirmed ahead of node A. The outcomes of these experiments are delineated in Fig. 9. Our observations indicate that with an escalation in the number of forged node records, there is a corresponding increase in node B's connections. If we set aside random variations, a positive correlation emerges between the number of block and transaction confirmations and the number of forged records. This indicates that the practice of node record forgery can, to some extent, expedite the confirmation of blocks and transactions.

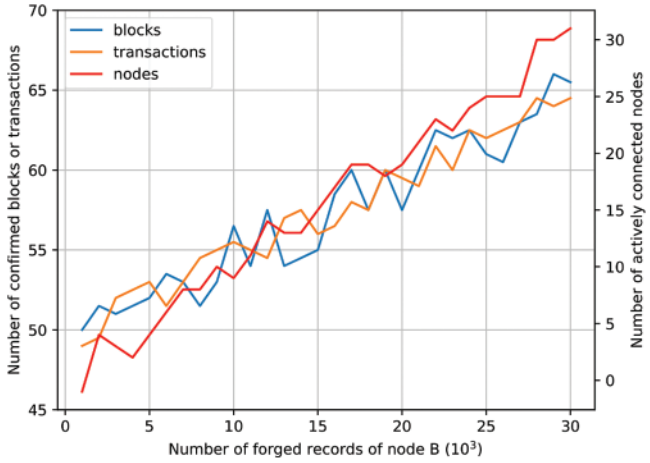

Figure 9: Changes in the number of blocks and transactions confirmed

5.3 Centralized Forgery

Centralized forgery, which is also commonly encountered within the network, is in stark contrast to the previously discussed random forgery. Table 3 details the prefix distribution of node records originating from the IP address 165.xx.xx.xx. In total, 20,231 node records are traced back to this IP address. Under uniform distribution, each prefix of length 2 would have an average of 79 occurrences.

Upon analysis, it is evident that the node ID distribution in this forgery type is extremely uneven, with the majority heavily concentrated on a few specific prefixes.

Table 3: Table captions should be placed above the tables

Prefix	Number of records	Deviation from mean
18	1915	1836
1C	1888	1809
47	1	78
63	2079	2000
69	2050	1971
86	2096	2017
B2	1959	1880
B3	2020	1941
C0	2022	1943
D7	2051	1972
FF	2150	2071

The potential motivation behind this behavior is to target specific nodes. When a local node decides on a target node to establish a connection with, it defaults to connect to the node situated in the first bucket, implying that proximity enhances the likelihood of an active connection. An attacker could potentially generate a multitude of node records close to the target node and actively broadcast them to the target. As a result, the known nodes around the target gradually get supplanted by these maliciously forged nodes. If the attacker ultimately gains control over all the target node's external connections, they could halt the forwarding of new blocks or alter the contents of new blocks without detection by the target node. This method of attack bears resemblance to the eclipse attack [8].

To validate the effectiveness of the attack method, we utilized the previously mentioned simulation environment, designating node A as the target and node B as the attacker. We set the maximum number of connections for node A to 50, while node B generated 10,000 forged node records, storing the 100 closest to node A in the node database. We allowed node A to join the network as normal and recorded the relationship between the number of forged nodes to which node A was connected and the number of new blocks. To simulate real-world network fluctuations, we conducted three experiments. Each time 1, 2, or 5 blocks were produced, a normal node was randomly selected from node A's peer nodes to disconnect. The experiment results are displayed in Fig. 10. The results suggest that as the block height increases, all nodes connected by node A eventually become forged nodes produced by node B. Moreover, the frequency at which normal nodes disconnect correlates directly with the speed at which the number of forged nodes increases.

This kind of forged record also originates from nodes we term "Masquerade Nodes". These nodes do not actively broadcast node records to the outside world. However, when other nodes initiate a query to them, they promptly return several node records. These records are close in terms of node ID to the querying node and include the masquerade node's own address. As these types of nodes never maintain a stable node ID in the network, we refer to them as "Masquerade Nodes". As an example, consider a masquerade node we discovered at the address 50.xx.xx.xx, where over 99% of its familiar nodes are combinations of its own IP address and different node IDs. If a node selects a masquerade

node as the seed node, it becomes completely at the mercy of the masquerade node, only able to receive data that the latter is willing to send.

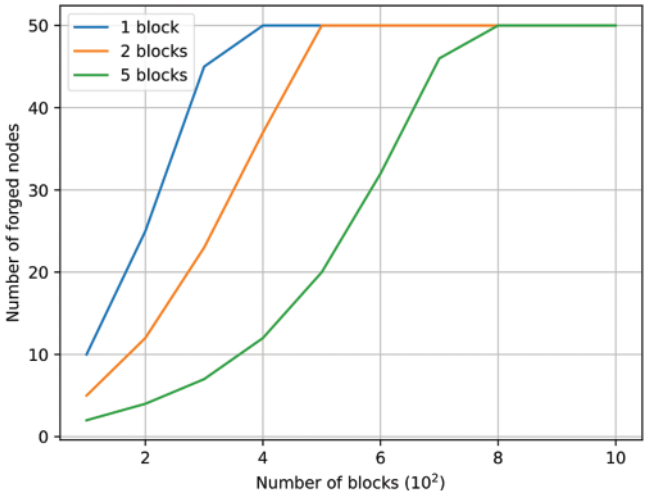

Figure 10: Changes in the number of connection forged nodes

5.4 Coping Strategy

Unlike application layer security issues, such as those concerning smart contract code [25], which requires meticulous code-writing on the part of users, issues on peer-to-peer networks necessitate solutions via a more foundational approach.

One potential solution is to increase the generation cost of forged node records. We could employ a mechanism similar to proof-of-work for the generation of node records. According to experimental results derived from the previous simulation environment, if the time cost of generating a node record is increased to one second each time, the damage caused by forged records could be significantly reduced. In a large-scale real environment, attackers would need to forge a larger number of node records. By increasing the time cost of generating a node record, we can elevate the difficulty of forgery, thereby enhancing prevention capabilities.

Additionally, the local node should designate a fixed connection node as a trusted node and limit the number of peer nodes from the same network segment. Moreover, operations engineers should maintain continuous vigilance over the connection status of peer nodes, and should not rely solely on the client's automatic routing algorithm.

6 Conclusion

In this paper, we have introduced a novel detection method for the Ethereum network that capitalizes on the characteristics of the node discovery protocol, enabling us to obtain previously unanalyzed node acquaintance relationships from the node database.

Our analysis of experimental results reveals prevalent forgery behavior in the current Ethereum network. Currently, almost half of all records are maliciously forged. We have categorized the observed forgery into two types: random forgery and centralized forgery. Specifically, we have identified a special case of masquerade nodes. And then we have proposed several possible reasons for these phenomena and verified these conjectures through experiments.

In future research, we aim to discover a more accurate method for detecting the number of node connections, which would enable us to more accurately discern the roles of various active nodes within the network. Additionally, it is imperative to quantitatively analyze the impact on communication efficiency and network security caused by forged node records and devise strategies to effectively counter the myriad malicious activities that are rampant in blockchain peer-to-peer networks.

Acknowledgement: An earlier version of this paper was presented at the International Conference on Computational & Experimental Engineering and Sciences 2023.

Funding Statement: This work is supported by the National Key Research and Development Program of China (No. 2020YFB1005805), Peng Cheng Laboratory Project (Grant No. PCL2021A02), Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies (2022B1212010005), Shenzhen Basic Research (General Project) (No. JCYJ20190806142601687), and Shenzhen Stable Supporting Program (General Project) (No. GXWD20201230155427003-20200821160539001).

Author Contributions: The authors confirm their contribution to the paper as follows: study conception and design: Y. Liu, Z. Lin, X. Wang; data collection: Z. Lin, Y. Zhang; analysis and interpretation of results: Z. Lin, Y. Zhang; draft manuscript preparation: Y. Liu, Z. Lin, Y. Zhang. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: All data incorporated in this study can be accessed by contacting the corresponding author upon request.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Ferdous, M. S., Chowdhury, M. J. M., Hoque, M. A. (2021). A survey of consensus algorithms in public blockchain systems for crypto-currencies. *Journal of Network and Computer Applications, 182,* 103035.
2. Lin, D., Chen, J., Wu, J., Zheng, Z. (2022). Evolution of ethereum transaction relationships: Toward understanding global driving factors from microscopic patterns. *IEEE Transactions on Computational Social Systems, 9(2),* 559–570. https://doi.org/10.1109/TCSS.2021.3093384
3. Franzoni, F., Salleras, X., Daza, V. (2022). AToM: Active topology monitoring for the bitcoin peer-to-peer network. *Peer-to-Peer Networking and Applications, 15(1),* 408–425.
4. Ethereum peer-to-peer network protocols specifications. https://github.com/ethereum/devp2p
5. Kim, S. K., Ma, Z., Murali, S., Mason, J., Miller, A. et al. (2018). Measuring ethereum network peers. *Proceedings of the Internet Measurement Conference 2018*, pp. 91–104. New York, USA. https://doi.org/10.1145/3278532.3278542
6. Etherscan: Token tracker. https://etherscan.io/tokens
7. Wang, S., Yuan, Y., Wang, X., Li, J., Qin, R. et al. (2018). An overview of smart contract: Architecture, applications and future trends. *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 108–113. Changshu, China, IEEE. https://doi.org/10.1109/IVS.2018.8500488
8. Marcus, Y., Heilman, E., Goldberg, S. (2018). *Low-resource eclipse attacks on Ethereum's peer-to-peer network*. Cryptology ePrint Archive. https://eprint.iacr.org/2018/236
9. Johnson, B., Laszka, A., Grossklags, J., Vasek, M., Moore, T. (2014). Game-theoretic analysis of DDoS attacks against bitcoin mining pools. *International Conference on Financial Cryptography and Data Security*, Berlin, Heidelberg, Springer. https://doi.org/10.1007/978-3-662-44774-1_6

10. Shafiq, M., Tian, Z., Bashir, A. K., Du, X., Guizani, M. (2021). CorrAUC: A malicious Bot-IoT traffic detection method in IoT network using machine learning techniques. *IEEE Internet of Things Journal, 8(5),* 3242–3254.
11. Tian, Z., Li, M., Qiu, M., Sun, Y., Su, S. (2019). Block-DEF: A secure digital evidence framework using blockchain. *Information Sciences, 491,* 151–165.
12. Qiu, J., Tian, Z., Du, C., Zuo, Q., Su, S. et al. (2020). A survey on access control in the age of Internet of Things. *IEEE Internet of Things Journal, 7(6),* 4682–4696.
13. Tian, Z., Luo, C., Qiu, J., Du, X., Guizani, M. (2020). A distributed deep learning system for web attack detection on edge devices. *IEEE Transactions on Industrial Informatics, 16(3),* 1963–1971.
14. Shafiq, M., Tian, Z., Sun, Y., Du, X., Guizani, M. (2020). Selection of effective machine learning algorithm and Bot-IoT attacks traffic identification for Internet of Things in smart city. *Future Generation Computer Systems, 107,* 433–442.
15. Chen, R., Shu, F., Huang, S., Huang, L., Liu, H. et al. (2021). BIdM: A blockchain-enabled cross-domain identity management system. *Journal of Communications and Information Networks, 6(1),* 44–58.
16. Ethereum node discovery protocol. https://raw.githubusercontent.com/ethereum/devp2p/master/discv4.md
17. Ethereum rlpx transport protocol. https://raw.githubusercontent.com/ethereum/devp2p/master/rlpx.md
18. go-ethereum source code. https://github.com/ethereum/go-ethereum/blob/master/p2p/discover/v4_udp.go
19. Maeng, S., Essaid, M., Lee, C., Park, S., Ju, H. (2021). Visualization of ethereum P2P network topology and peer properties. *International Journal of Network Management, 31(6),* e2175.
20. Maeng, S. H., Essaid, M., Ju, H. T. (2020). Analysis of ethereum network properties and behavior of influential nodes. *2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 203–207. Daegu, Korea (South), IEEE. https://doi.org/10.23919/APNOMS50412.2020.9236965
21. Etherscan: Node tracker. https://etherscan.io/nodetracker
22. Ethereum network hash rate. https://etherscan.io/chart/hashrate
23. Gleich, D. F. (2015). Pagerank beyond the web. *SIAM Review, 57(3),* 321–363.
24. Geolite2 free geolocation data. https://dev.maxmind.com/geoip/geolite2-free-geolocation-data?lang=en
25. Wang, Z., Jin, H., Dai, W., Choo, K. K. R., Zou, D. (2021). Ethereum smart contract security research: Survey and future research opportunities. *Frontiers of Computer Science, 15(2),* 1–18.