

Bitcoin Network: Using Iterative Algorithms for INV-Message Based Topology Inference

Marvin Woller

*DSN Research Group, KASTEL Institute
Karlsruhe Institute of Technology
Karlsruhe, Germany
marvin.woller@woller-it.de*

Matthias Grundmann

*DSN Research Group, KASTEL Institute
Karlsruhe Institute of Technology
Karlsruhe, Germany*

Hannes Hartenstein

*DSN Research Group, KASTEL Institute
Karlsruhe Institute of Technology
Karlsruhe, Germany*

Abstract—Decentralized flooding Peer-to-Peer networks are the basis of world wide used crypto currency systems as Bitcoin. Topologies of such systems are in general not public available because one of the main security mechanisms is hiding them. In this paper we show that methods for inferring topologies already exist in literature. We also propose new methods with the goal to show that observing transaction and block propagation and applying iterative algorithms could break the Security-By-Obscurity concept. We use simulation to test the basic concepts of our algorithms and to determine parameters and show in the end, using real world Bitcoin data, that topology inference is feasible.

Index Terms—Bitcoin, P2P, Topology, Inference, Blockchain

I. INTRODUCTION

Since 2009 when Bitcoin was released, the Bitcoin network has become one of the most famous and intensively used Peer-to-Peer (P2P) networks in the world [1], [3]. With an increasing amount of users every day who exchanges large amounts of money [14], achieving a high security level becomes more and more important. In the last few years there was a lot of research in extensively studying crypto currency systems and their related network infrastructures [16] - [18]. In case of Bitcoin, the developers included a lot of new concepts in the Bitcoin protocol to improve security and resilience [4]. As we show later on, there is a whole number of attacks that could possibly threat the security of a particular user or even damage a whole crypto currency system [6]. While hiding information like the topology of the underlying P2P is state of the art approach for achieving security, current research shows that topologies could be inferred [5]. With knowledge of the topology of P2P networks attackers can perform more powerful and directed attacks on specific and highly relevant points in the network [14]. On the other hand, if developers have knowledge or are able to monitor the topology actively improving the resilience and security of such a network becomes more feasible and countermeasure methods against new attack vectors could be applied in time [14].

In this paper we therefore first investigate some of the existing solution and afterward propose new iterative approaches for inferring topology of P2P networks. All of our methods are based on the investigation of monitored inventory (INV) messages [11]. At the end we show that we are able to perform

topology inference on the real Bitcoin network with precision and recall of around 23%

II. MOTIVATION

The relevance of topology inference has been shown in literature many times. While topologies has direct influence in effectivity of information propagation, knowledge about the topology could be use to improve the performance [14]. Knowing paths and the number of connections, propagation delays and latencies could be reduced what improves the usability of the networks.

Before we focus on the actual project of Bitcoin network topology inference we give a short introduction into peer-to-peer networks and the meaning of network topology and we consider the underlying terminology to gain understanding for the Bitcoin networks setup and behavior and the basic concepts of topology inference.

III. FUNDAMENTALS

A. Peer-to-Peer Networks

In general, P2P networks are common used network communication infrastructures for highly distributed internet services. In contrast to client-server architectures, P2P network operate with equal participating members without hierarchical organization structures [7]. P2P networks exist in many different forms and constellations. A special case are flooding P2P networks. These networks use gossiping protocols for information exchange to achieve knowledge equality between different peers. Gossiping in this case means that messages that arrive at any peer are broadcasted to all connected neighbors of this peer that still don't contain the message. This procedure is repeated for every peer that receives such a message [5].

1) *Bitcoin Network*: Because we focus in this paper on the Bitcoin network we take a closer look on the structure of the Bitcoin network and it's flooding mechanism. The Bitcoin network is a world wide distributed decentralized large scale P2P network [5] consisting of over 10000 nodes [8].

The types of nodes of the networks can be divided into two groups. On the one hand reachable peers are the main part of the network and make up to 90% of it [cite barcelona]. These reachable peers maintain the majority of the connections of the network and are therefore much more relevant for any kind of

message propagation [cite]. On the other hand the network includes non-reachable peers that we can not connect to. The unreachable peers form only a small part of the network. Because performing any kind of analysis is much easier on reachable peers we only focus on the reachable part of the network in this paper.

The main purpose of the Bitcoin P2P network is forwarding transactions and Blocks. If a transaction is triggered by a user and started at a specific node, this transaction needs to be distributed through the whole network to all participating peers. Same thing for blocks. If the transaction is included in a block, this block also needs to be distributed to keep the network in a consistent state.

B. Flooding

In order to propagate messages like transactions or blocks across the network, Bitcoin implements a gossip-based flooding protocol. This protocol consists of three steps. First, if a node has received a new message, it sends a inventory message (INV) with a hash value to all neighbors telling them that some new information is available. The informed neighbors then can decide in a second step whether they request this new information or not, depending on their current inventory. If they already have the information they do nothing but if it is missing they request it using a GETDATA message. In the last step the message is sent using a TX or BLOCK message [15].

C. Monitor

A special member of the Bitcoin P2P network is the monitoring node. It connects itself to almost all reachable nodes to receive INV messages. This node is passive and only listens to the network and never sends GETDATA or INV messages to its neighbours [5].

D. Topology

A network topology is the physical and logical arrangement of nodes and connections in a network. It describes the arrangement of networks and the relative location of traffic flows.

The network topology plays a major role in how a network works and has a direct effect on the functionality. Choosing the right topology can help increase performance, resilience and robustness of a network. A properly chosen and maintained network topology increases energy efficiency and data transfers rates. Latency could be reduced by routing messages over shortest paths and knowing highly connected nodes reduces the risk of network faults due to the loss of an important single-point-of-failure node [14]. A known network topology makes it easier to locate faults, troubleshoot issues, allocate network resources and to help diagnosing and developing the network in general [5], [19].

1) *Topology Inference*: Topology Inference has the goal to uncover the topology of a network.

In case of Bitcoin, this is difficult because Bitcoin network is deliberately designed to hide its topology from onlookers.

Knowledge of the topology is not in itself a vulnerability, although it could conceivably help an attacker performing targeted eclipse attacks or to deanonymize transaction senders [2].

IV. RELATED WORK

The basic idea of performing topology inference on P2P networks such as Bitcoin network is not totally new and a lot of research have been done so far.

In 2016 Neudecker et al. [5] proposed an approach for Bitcoin network topology inference by using detailed timing analysis of INV announcements and a probability function to classify peers as being adjacent or not. In a first step they applied their method on the basis of an analytical propagation delay model and validated it afterwards with a simulation. In a second step they applied their method using an empirical propagation delay model to the real Bitcoin P2P network. The results of the work of Neudecker et al. show that gathering adjacency information only by using timing differences of the received INV announcements is generally feasible. Besides the theoretically simulation based approached they showed that their system is able to infer the topology of the real Bitcoin network with an precision and recall of around 40%. With that results they came to the conclusion that the ability of performing efficient topology inference leads to new chances in terms of improving power of the network and the propagation of messages but also gives attackers advantages for damaging the network or compromising the users privacy.

Building upon the work of Neudecker et al., Daniel et al. [15] developed a method for passive topology inference exploiting the block relay mechanism. In their work they focused merely on inferring the links of miners to their connected neighbor nodes. Because they only used the propagation data of blocks, their method is independent from active timing inference countermeasures like trickling or diffusion that take place when propagating transaction.

To infer the topology of the ZCash network they used latency differences in monitored INV announcements. Their basic assumption was that if latency values with "low enough" value were observed the nodes are considered as being adjacent.

To evaluate and verify their model Daniel et al. went also a two way strategy consisting of a simulation and real-world approaches. Using the simulation they determined the parameters of the model.

When testing the model in real-world experiments it achieved 50% precision and 82% recall, proving applicability in the real world.

Daniel et al. provided a good solution for topology inference that could be applied to a lot of crypto currency systems. In our work we will also try a similar approach using a simulation first to obtain good model parameters and apply it then to the algorithms using real-world Bitcoin network data to check its applicability. We will also construct a similar latency based approach and try to validate it on real Bitcoin network.

V. SIMULATION

If P2P networks rise to huge size with many peers and many random connections, making statements gets difficult. If the network is even a flooding one as Bitcoin network, it is even more complicated. Performing analytical analysis using closed form solutions gets nearly impossible and empirical analysis is difficult in terms of data storage and execution time. The goal therefore was to get realistic insights into a Bitcoin like P2P network that has nearly the same properties and behavior as the real one but is highly reduced in size to perform empirical analysis faster but with similar and scaleable results.

Using a simulator and simulation [9] gained values helps us to improve the understanding of the P2P networks behavior, of the flooding processes, and allows us to easily perform experiments.

A. P2P Simulator Implementation

In the following we propose our implementation of a discrete event simulation (DES) [10] based P2P network simulator. The simulator is totally object orientated written in python. It can generate a network according to given information like number of nodes and simulates the message propagation of a given number of messages. Message propagation always starts from a randomly selected node.

Before the simulation of message propagation can be started, the network needs to be created. This process of network creation can be separated into tree steps. First, a list of N random and independent IP addresses is created. N in this case is a given parameter that represents the total number of nodes. With that IP list and another given parameter p a adjacent matrix with size $N \times N$ is created. The adjacency matrix is implemented as a dictionary that holds the IP addresses as keys. The parameter p represents the value of 1-fillings of the matrix and with that the amount of connections between nodes. This means that in a network generated from a matrix with $p=0.5$ every node have connections to 50% of the other nodes in the network. For the special case of simulation of the Bitcoin network every node has eight neighbours.

To perform simulation, it is necessary to generate from the adjacency matrix the nodes of the object oriented network with a special object as event handler for DES. Every node is a object with a send and receive method for messages and contains a list of all connected neighbors.

1) *Messages*: To simulate the propagation of transactions or blocks, nodes are randomly selected with a equal distribution and the send method of the selected node is called handing over a new transaction object. The method for generating transactions takes a parameter n as input that represents the number of transaction that will be generated and released.

If a node receives a new message which the node has not already received, it calls it's own send method.

This send method contains a loop iterating over the list of neighbors. For every neighbor accepting this message the event handler is called with this transaction, the calculated receiving time for the neighbor and the neighbor's IP address.

2) *Event handler*: The event handler is the central object of DES and mainly used for transaction distribution. All nodes that send transactions to their neighbours do not send them directly but call the event handler's *addEvent* method to generate and append a new event to the execution stack. An event consists among other things of an IP and a timestamp.

With an internal clock, the event handler always executes the events with timestamps lower or equal to the internal clock.

3) *Monitor*: As in the real world setup [11] the simulator contains a monitoring node.

With each received message an INV message is sent to the monitor. The Monitor stores all INV messages in a list that could later be used for topology inference.

VI. TOPOLOGY INFERENCE

In the following we propose three iterative algorithms for topology inference on flooding P2P networks. All inference models use INV messages grouped by hash and ordered by timestamp as input and generate a adjacency matrix as output.

TABLE I
ALGORITHM DATA INPUT SCHEME

Hash	INV Data	
	Timestamp	IP
hash1	t_1^{h1}	ip_1^{h1}
	...	...
	t_n^{h1}	ip_n^{h1}
hash2	t_1^{h2}	ip_1^{h2}
	...	...
...		

A. INV Order based topology inference

To extract the topology from a totally unknown network, multiple approaches exist. Because every network has different behavior and usable assets, the algorithms are hardly different and in many cases not usable. As Neudecker et al. have shown, topology inference on the Bitcoin network is generally feasible by analyzing timing differences of centrally observed INV announcements. This approach works actually well and could be use to gain information from the network structure but requires some effort for calculation and is broken by the introduction of diffusion instead of trickling [15]. In the following we propose a new algorithm for topology inference that does not rely on exact timing analysis but gains information from the order of received INV announcements. The timing aspect in our algorithm is only necessary to generate an ordered List of the INV announcements but the investigation of exact time deltas as in [5] is not required. Based on the order of the INV announcements the nodes are being rated as adjacent by an decreasing probability function.

Step 1: Let M_{adj} be an empty adjacency matrix with size $N \times N$, where N denotes the total number of all Peers available in the network (related to when it was observed). Further, we have a list L of all INV announcements with the form $[(timestamp, hash, ip)_1, \dots, (timestamp, hash, ip)_n]$.

Step 2: The INV announcements in L are sorted as they were received by the monitor. Because the inference algorithm requires as input in each iteration a selected hash with a list of pairs of its corresponding IP addresses and timestamps, grouping and sorting L is necessary. To change L into a suitable format, we introduce the dictionary D and fill it with all elements that L contains. The hash values in L are used as keys in D and the IP addresses and timestamps are putted as tuples in a list and set as value of $D[hash]$ with $D[hash] = \text{List}((\text{timestamp}_0, \text{ip}_0), \dots, (\text{timestamp}_m, \text{ip}_m))$.

Step 3: With D it is possible to perform the iterative topology inference algorithm. The first iteration loop is performed for all hash values in D , meaning we iterate over all lists in D . For each list $L_{hash} = D[hash]$ a second round of iteration is performed. In this second round the probability function is applied to the elements in L_{hash} . We call this second iteration “inner iteration”.

1) *Probability Function:* To model the probability that two nodes are being adjacent we use for the nodes in each list L_{hash} a probability function F . F assigns each node a probability value of being adjacent to all following nodes according to the order in the (by timestamp) sorted list. We assume that if a INV messages is received from the monitor at a given timestamp t_x from IP_x , the probability that this node is adjacent to the node with IP_{x+1} that send a INV message received at timestamp t_{x+1} is higher than being adjacent to a node with $IP_{x+k}(t_{x+k})$ for $k > 1$. For Example: The probability of IP_0 is adjacent to IP_1 is bigger than the probability of IP_0 is adjacent to IP_2 . In general the probability of being adjacent decreases with the order related distance of a node to the other following nodes in L_{hash} .

To model the probability decay we use as F different functions as shown in figure 1.

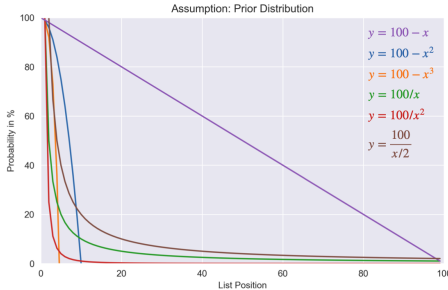


Fig. 1. Decay Functions

The main parameter that is taken by each function is the number of the current inner iteration that is equivalent to the position in L_{hash} relative to the starting node. Additionally, the position of this starting node increases with each inner iteration by 1. In figure 2 we illustrated the process of inner iteration. In this example the network contains three peers and one transaction is sent from peer 1 (P_1) to peer 2 (P_2) and from peer 2 to peer 3 (P_3).

2) *Adjacency Matrix:* The main result of the algorithm is a filled matrix M_{adj} where each entry (i, j) contains the

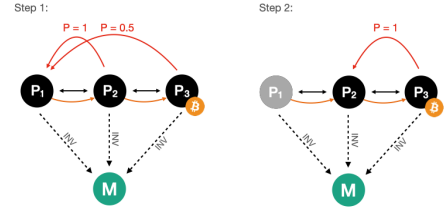


Fig. 2. Inner Iteration

probability of node with IP_1 being adjacent to node with IP_j . M_{adj} is filled iteratively. With each round of inner iteration the probability values are added to the corresponding field in M_{adj} .

After the process of inner iteration was performed for all available hash values, meaning the outer iteration process has finished, M_{adj} is finally filled and could be further processed. At this point M_{adj} contains in each field (i, j) the sum of the probability values that i is adjacent to j . To decide which nodes are decided as being adjacent it is necessary that M_{adj} is normalized and each entry is mapped to 0,1 with 0=not adjacent and 1=adjacent. To normalize M_{adj} , each entry is divided by the maximum probability value in M_{adj} . Each entry in M_{adj} is scaled to $[0, 1]$ and could be interpreted as the direct adjacency probability. With a threshold T with $T \in [0, 1]$, each entry in M_{adj} is compared to and decided as being adjacent if $(m_{ij} > T)$ or not if $(m_{ij} < T)$.

B. Latency based topology inference

The second methods we propose for inferring the topology is based on the work of Daniel et al. [15]. In contrast to our previously described method the following algorithms uses besides the order of the INV messages also the latencies of the INV messages between different peers. In this approaches we fill M_{adj} not with probability assumptions from a prior probability function relative to the plain order as before but with the latencies between the INV messages send by two peers. This takes place in an iterative process. For each hash the latencies are calculated as the difference in time of two arrived INV messages. This timing difference is then written to the adjacency matrix according to the corresponding IP addresses (IP_a, IP_b) and given rules. The base rule for updating M_{adj} is that if the calculated value is lower than the value in M_{adj} for (IP_a, IP_b) then it is replaced by the new and lower one. This results in a matrix M_{adj} that stores the lowest found latency values.

According to [15] we assume that if a latency value is “small enough” two nodes are considered as being adjacent.

This latency based approaches could be performed in different ways depending on how the latencies are exactly calculated.

1) *1st Approach: Order dependent:* As before, the starting point is a dictionary D of hashes as keys and lists L_{hash} with the corresponding data (timestamp and IP address) from the INV messages belonging to the hash. For each hash

and its corresponding list L_{hash} the following algorithm is performed. L_{hash} is a by timestamp ascending ordered list of INV messages. While an iteration from the beginning to the end of L_{hash} is performed, the time differences for all pairs are calculated, as shown in figure 3.

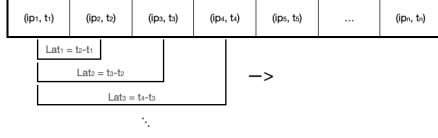


Fig. 3. Calculation of Latencies

Every calculated time difference is then written to M_{adj} according to the updating rules.

Punishment: When using by timestamp ordered lists, we add a punishment value to the latency to model the increasing uncertainty when messages propagate long time through the network. If we consider the network as a tree structure where the message starts at the root node. It is possible that in a deeper layer two nodes on different paths that are not directly connected receive the message at the same time and send the INV message simultaneously to the monitor. By just looking at the ordered INV list or the latency value these nodes are being considered as adjacent even if they not. To avoid this we add a punishment value.

2) *2nd Approach: Order independent:* In the previously described methods we use for each hash a ordered list of INV announcements. When performing the latency calculation the order of the list has in the previous approach an effect because only INV messages that arrived directly after each other are being considered pairwise for the calculation of the latency. Now we want to break the structure of this ordered list and compare all INV messages that are received for the corresponding hash pairwise together. As before every calculated latency is then written to M_{adj} based on the updating rules.

VII. EVALUATION

In the following section we show the applicability of our previously proposed topology inference models. This section is divided into two parts. In the first part we use INV data generated with simulation and evaluate and verify the models with this data. In the second part we apply the models to real world data of the Bitcoin network collected by the monitor.

A. Metrics

To evaluate the models we use the F_1Score score. The F-score is defined as a harmonic mean of precision (P) and recall (R) [12].

$$F_1 = \frac{2PR}{P + R} \quad (1)$$

B. Simulation

Using simulated networks gives us the ability to inspect the quality of our topology inference models for different networks by adjusting their parameters as network size/number of nodes,

the amount of used transactions and different propagation techniques.

The size of the network has a large effect on the inference quality. With the simulator we generated networks different in size with 10 up to 1000 peers and performed topology inference with the models using a static amount of 500 transactions. As shown in figure 4 the F_1Score score drops if the network size increases.

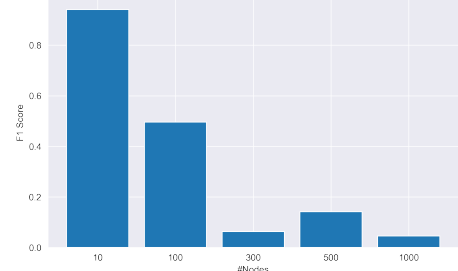


Fig. 4. Size of network vs. quality (Order based approach)

The quality of the models suffer if the networks get bigger. A basic countermeasure against this quality loss is using more data. When using iterative algorithms the result is updated with each step. This means that each iteration extracts information from the data, resulting in that more data leads to better results and higher quality. We proved this empirically as shown in figure 5.

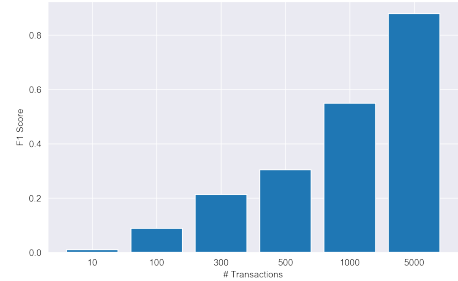


Fig. 5. Amount of data vs. resulting quality (Order based approach)

1) *Topology Inference Models: Hyperparameters:* With help of the simulation we have shown that the nature of the network has an effect on the topology inference. Now we focus on the inference algorithms itself and their hyperparameters.

The following values resulting from a simulated network with 1000 peers and 1000 propagated transactions. As mentioned in section VI-A1 we used different functions to describe the probability decay while performing the inner iteration. Figure 6 shows that these different functions result in different F_1 scores and it proves empirically that the decay function is part of the main optimization problem.

As figure 6 shows using $F = \frac{100}{x}$ as decay function leads to the best results.

Further, the threshold T that is used to decide if two nodes are adjacent is something to optimize. With the help

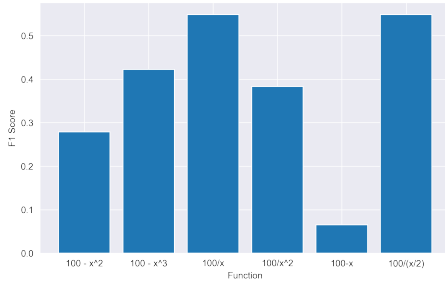


Fig. 6. Decay functions (Order based approach)

of simulation we found out empirically that 40% is a good value for T .

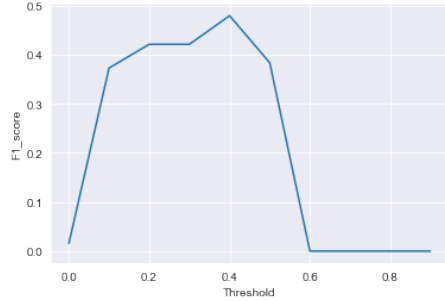


Fig. 7. Decision threshold (Order based approach)

Note: To not overload this paper we only considered in the last section results for determining parameters for the order based approach. The process for determining parameters for the latency based approach is analog.

C. Real World Data

In the following we evaluate the models on the real Bitcoin network data using INV messages collected by the monitor. We additionally use the connection data of two peers that are part of the Bitcoin network and try to verify the quality of our models. The neighbours of this peers are known and we use them as ground truth to compare their connections against the results of the topology inference models. We use again the F_1 score as metric but also look at precision and recall.

1) *Results:* In the following we use INV messages collected by the monitor for one day on June 13th 2022. This dataset contains approx. 250K INV messages propagated by 9216 peers. The nodes we use for verification have the IPs 193.169.39.103 and 2001:7c0:2310:0:f816:3eff:fed6:8580.

Topology Inference Results			
Model	Precision	Recall	F_1 Score
Order	0.273	0.2	0.231
Latency 1 st Appr.	0.191	0.170	0.18
Latency 2 nd Appr.	0.088	0.132	0.105
Latency 1 st Appr. + Pun.	0.196	0.170	0.182

Table VII-C1 shows the results of the evaluation. The best of the three models is the order based approach with an F_1

score of 23%. The latency based approaches perform slightly worse than the order based one. Even if F_1 scores are lower than others in literature, this results show that the order based as well as the latency based approach could be used to gain information from an actually hidden topology.

D. Temporal effects

We always have to keep in mind that open distributed P2P networks change their topology dynamically in relation to time. Huge P2P networks as Bitcoin network are not static and nodes as well as connections do appear or disappear in a unpredictable way. In Bitcoin network users are free to add or remove the nodes they are operating. If new nodes are added by user, the nodes advertise their selves in the network and get dynamically integrated. If a node is removed, the neighbor nodes are able to find substitutes and establish new connections. This always effects the topology just only a bit but if such events happen many times in different locations the global topology changes.

Conversely, that means that the accuracy of inferred topologies decreases with time. Even if we infer the topology of such a network with high accuracy, the structure of the network changes with ongoing time and the connections that were inferred vanish or new connections emerge and the model doesn't match the real topology anymore. If we finally want to gain information from an inferred topology its therefore important that the model is not too old.

1) *Problems:* When applying iterative algorithms on topology inference problems, a lot of data is necessary. Generally the results gets more exact when we use more data. The problem at this point is, that collecting data, as we did e.g. with the monitoring node, is limited by used resources and time. If we train the inference model with data, which is collected over a long period of time, we get some noise resulting from the changes of the real network topology while the data collection was performed. The optimization of the data collection period needs definitely being further investigated and we consider it as a great field for future work.

VIII. CONCLUSION & FUTURE WORK

In this paper we have shown that topology inference on P2P networks and especially on Bitcoin network is generally feasible. In literature there exist a lot of promising approaches and we showed that our models perform well on small networks. Even if our models suffer from the size of real Bitcoin network we still achieve F_1 scores of around 23%. This seems low but we need to keep in mind that this knowledge could still help attackers to identify important points in the network and improve attacks, or it could even help to improve routing and reduce latencies.

Future research should on the one hand focus on improving the algorithms by trying other probability decay function or in case of the latency based approach, trying other punishment functions. On the other hand finding the corresponding thresholds is also an ongoing optimization problem. Maybe a combination of the models could also lead to better results. In

general more data and longer training time is required to get more exact models.

Finally we conclude that there exist a lot of possibilities to break the Security-by-Obscurity mechanism that is used in Bitcoin network. Inspired by Kerckhoffs's principle [13] hiding information like the topology is not a good solution for security. With enough effort and resources topologies could be well inferred. From our point of view keeping topology information public has more benefits than disadvantages. With a known topology routing and message propagation could be improved and latencies could be reduced. In times where the popularity of using crypto currencies rises hardly it seems advantageous to improve the usability of such systems and focus in terms of security on other methods that are not interfering with the usability and network power.

REFERENCES

- [1] Nakamoto, S. (2008) Bitcoin: A Peer-to-Peer Electronic Cash System
- [2] Delgado Segura, Sergi & Bakshi, Surya & Pérez-Solà, Cristina & Litton, James & Pachulski, Andrew & Miller, Andrew & Bhattacharjee, Bobby. (2018). TxProbe: Discovering Bitcoin's Network Topology Using Orphan Transactions.
- [3] Chohan, Usman W., A History of Bitcoin (February 5, 2022). Available at SSRN: <https://ssrn.com/abstract=3047875> or <http://dx.doi.org/10.2139/ssrn.3047875>
- [4] Fanti, Giulia C. and Pramod Viswanath. "Deanonymization in the Bitcoin P2P Network." NIPS (2017).
- [5] T. Neudecker, P. Andelfinger and H. Hartenstein, "Timing Analysis for Inferring the Topology of the Bitcoin Peer-to-Peer Network," 2016.
- [6] Arthur Gervais, Ghassan O. Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. 2016. On the Security and Performance of Proof of Work Blockchains. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16). Association for Computing Machinery, New York, NY, USA, 3–16.
- [7] Galuba, W., Girdzijauskas, S. (2009). Peer to Peer Overlay Networks: Structure, Routing and Maintenance. In: LIU, L., ÖZSU, M.T. (eds) Encyclopedia of Database Systems. Springer, Boston, MA.
- [8] <https://coin.dance/nodes>
- [9] Stewart Robinson. 2004. Simulation: The Practice of Model Development and Use. John Wiley & Sons, Inc., Hoboken, NJ, USA.
- [10] Norm Matloff. February 13, 2008. Introduction to Discrete-Event Simulation and the SimPy Language
- [11] Neudecker, Till. (2019). Characterization of the Bitcoin Peer-to-Peer Network (2015-2018). 10.5445/IR/1000091933.
- [12] Sasaki, Y. (2007). "The truth of the F-measure"
- [13] Kerckhoffs, Auguste (January 1883). "La cryptographie militaire"
- [14] 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, 408-425.
- [15] Daniel, E., Rohrer, E., & Tschorsch, F. (2019). Map-Z: Exposing the Zcash Network in Times of Transition. 2019 IEEE 44th Conference on Local Computer Networks (LCN), 84-92.
- [16] Conti, M., Sandeep Kumar, E., Lal, C., & Ruj, S. (2018). A Survey on Security and Privacy Issues of Bitcoin. IEEE Communications Surveys & Tutorials, 20, 3416-3452.
- [17] Tapsell, J., Akram, R.N., & Markantonakis, K. (2018). An Evaluation of the Security of the Bitcoin Peer-To-Peer Network. 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 1057-1062.
- [18] Reid, F., & Harrigan, M. (2011). An Analysis of Anonymity in the Bitcoin System. 2011 IEEE Third Int'l Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third Int'l Conference on Social Computing, 1318-1326.
- [19] Miller, A.K., Litton, J., Pachulski, A., Gupta, N., Levin, D., Spring, N., & Bhattacharjee, B. (2015). Discovering Bitcoin's Public Topology and Influential Nodes.