International Journal of Electronics and Telecommunication Systems
Volume1, Issue 04 2025

Airdrop Sybil Attack detection framework supported by machine learning

Vaisuna[1], Sambhaji[2]
Department, ECE, Pune, India

Research Article

Date of Submission: 10-08-2025 **Date of Acceptance: 17-09-2025** **Date of Publication: 20-11-2025**

Abstract—A skilled hunter may earn thousands of dollars in tokens from a single airdrop, making airdrop Sybil assaults a profitable endeavor. In this situation, Sybil assaults are not always wanted by projects and are often seen as improper behavior by honest individuals, which may negatively impact a project's image. It is thus to be anticipated that detection methods for Sybil assaults would be continuously enhanced. This study presents a multistep structure. Its concept is to sort blockchain addresses and give each one a score that indicates whether it belongs to the Sybil or regular class. Topologies were classified using a graph isomorphism network, whose parameters were adjusted using a dataset that the authors labeled. In subsequent phases, the account clustering operation was performed using a DBSCAN. Each stage in the framework may be given arbitrary weights by users, which will decide how essential a step is to them and provide a different score for a particular address. A grid search strategy and a threshold after which the address is deemed Sybil were used to find the optimal weights. This article examined a collection of EOAs from the ZK sync rollup. Ultimately, 76% of the examined accounts were classified as Sybils. Compared to the official ZK sync eligibility list, we found 342 addresses that received airdrop tokens but were marked as Sybil by our solution.

***Keywords*—Blockchain, ZK sync, Airdrop, Sybil Attack Detection, and Graph Isomorphism Network**

I. INTRODUCTION

Project developers employ N blockchain airdrops as a tactic to attract users and boost on-chain activity. Usually, it entails providing incentives for doing certain project-related activities. Reward tokens are often sent (airdropped) to qualified addresses by airdrop producers. Developers may also design many airdrop events during the course of a project, such as the Opti-mism rollup. This is a typical practice among freshly launched companies. There is a clear distinction between proactive and retroactive airdrops, and they may have distinctive designs. In a totally retroactive architecture, the list of instructions to be eligible is shown after the network snapshot is obtained, and users gain tokens based on their previous behavior. Proactive design, on the other hand, makes it obvious what has to be done in order to get tokens. Both are somewhat vulnerable to so-called airdrop farmers. Airdrop Sybil attack, also known as airdrop farming, is a kind of assault in which one or a small number of organizations, referred to as farmers or hunters, establish several phony identities in an attempt to outwit the airdrop and significantly expand their token pool. Assets from an airdrop may sometimes be worth thousands of dollars to hunters [1]–[3]. A hunter was able to amass 933,365 Arbitrum tokens worth almost $1 million in 2023 [3]. Given that Sybil was one of several instances of a similar kind over the whole bitcoin market, this demonstrates the scope of the assaults.

Most people believe that airdrop farming is unfair to sincere users. After obtaining tokens, hunters often sell them quickly [4], which affects the market and may compel sincere token recipients to follow suit. Additionally, the project's general public relations may be discouraged by their effective presence. Instead of adding real value, hunters create artificial network activity only to mimic engagement and get rewards. By carefully crafting airdrop designs that include the use of proof-of-humanity, address blacklisting, self-reporting, or incentive designs that are difficult to spend, developers attempt to lessen Sybil assaults. Nonetheless, others contend that airdrop hunters may draw in some newbies and are not wholly bad [5]. The project should specifically take into account the strategies that should be used to address the Sybil attack phenomena.

Concurrently, the identification of Sybil accounts is another aspect of attack protection, and this study delves further into this subject. By examining clusters in a graph made from account-to-account transactions, Sybil attacks may be identified. Additionally, some transactional and structural patterns linked to Sybil attacks have been identified [6]–[8]. Additionally, the use of machine learning models to carry out detecting tasks is becoming increasingly widespread. Hunters, on the other hand, adjust to evade detection [1]. Sybil attacks may be carried out by well-organized groups of hardworking individuals who might be paid with the sole objective of airdropping the game and dividing the prize. Without the use of bots, a worker could manually and reliably establish, fund, and manage several accounts as well as execute intricate transactional patterns across blockchain addresses. Naturally, the total cost of carrying out such an assault would be more than that of a single hunter using bots.

The primary goal of Rollup, a Layer 2 solution, is to expand the Ethereum network. A blockchain called Rollup has a far larger

transaction volume per second.While Layer 1 (Ethereum) simply accepts evidence of state changes, Layer 2 networks process transactions. Rollups use a Layer 1 smart contract that uses a Merkle root to monitor the Layer 2 state [9].

Our investigation focused on the ZKsync Era network, one of the rollups based on zero-knowledge proofs. Compressed transactions from ZKsync blocks are sent to Ethereum in one batch. As a result, ZKsync costs are much less than those on Ethereum. This effectiveness has created a new possibility for hunters and made it a desirable choice for consumers. Sybil behavior is undoubtedly encouraged by lower transaction costs. The ZKsync team evaluated user behavior from the network launch on February 14, 2023, to March 24, 2024, using a retrospective airdrop model. For the drop, 17.5% of the total ZKsync token supply was set aside [10]. A ZKsync airdrop was made available to hundreds of thousands of wallets in March 2024. With their own set of eligibility requirements, the tokens were divided between two primary groups: network users and contributors like developers and researchers. Users were eligible for an address if they fulfilled at least one of seven requirements, including as engaging with ten or more non-token contracts, exchanging ten distinct tokens, or utilizing Paymasters at least five times. Additionally, how early and how much an address bridged to ZKsync determined how many tokens it earned. Accounts that demonstrated evidence of real human activity or a significant contribution received additional multipliers.

A. Paper Organization

The suggested framework's approach is described in Section II. The selection of machine learning algorithms has been given, along with a short description of each algorithm's key parameters. The findings are shown in Section III along with hypotheses about their potential origins. The chapter presents the outcomes of applying the framework to the ZKsync rollup dataset as well as neural network fine-tuning discoveries. The findings and subjects pertaining to the framework and field of study that the authors believe are significant are discussed in Section IV. The conclusion and future work are included in the last section V.

B. Related Work

There may not be much literature specifically pertaining to the detection of Sybil attacks on blockchain since the subjects addressed in the study are very new. General airdrop design ideas and other elements, including Sybil assaults or implementation techniques, are shown in [11]. An excellent summary of the field of airdrop research may be found in this paper. Another publication that describes real-world instances and discusses several airdrop systems is [12].

The authors of [8] provide a Sybil detection method based on identifying groups of accounts with comparable event histories in addition to looking for certain recognized Sybil-like topologies. The Hop protocol, which enables Layer 2 solutions, was the subject of their investigation. Additionally, examples of hunting behavior in the network were shown, such as how the sibilant graph

Topologies are created. The whole study is a fantastic contribution to the field of airdrop Sybil detection.

A Sybil attack detection mechanism for transactions using non-fungible tokens is given in articles [13] and [14]. They dubbed it ARTEMIS, and a neural network provided support for it. Additionally, the authors presented a comparison of metrics between their neural network and other machine learning techniques, including well-known graph neural networks like GIN. A glimpse of normal hunter trade patterns was also presented.

A game theory model that simulates the interactions between hunters and projects is presented by the authors in [15]. They demonstrate accessible tactics and suggest efficient reward schemes that enhance detection while minimizing operating costs. Additionally, Hop Protocol and LayerZero analysis served as the foundation for the study.

The authors of [16] provide a data-driven method to suggest a role-based taxonomy using ParaSwap as a case example. It aids in determining the community members' behavioral patterns. The study may be a useful addition to our work's overall discussion of airdrop Sybil attacks as it also discusses airdrop hunters.

II. METHODOLOGY

As seen in figure 1, we provide a multi-phase structure backed by machine learning techniques. Only a topology-based method might result in false positives because of the poisoning assault that hunters can execute. Depending on the context, there are many definitions of poisoning attacks in the blockchain space, such as [17] for harmful material put on the network and [18] for dust, zero, or fraudulent contract transfers. Phishing transfers are conducted from an attacker whose address is artificially created to resemble the victim's as much as possible, according to [18]. Addresses might be quite different in our environment. In order to trick detection systems by tapping into a different topology or, in the worst case, to do damage to others, a Hunter would send a transaction with very little value to unidentified accounts, marking honest people as Sybils. Therefore, by examining different account information, the framework attempts to tackle the challenge of detecting the majority of Sybils with the fewest misconstrued honest individuals. Its goal is to determine the Sybil score for every filtered blockchain address for a certain network lifespan. Each of the four methods used to generate the score yields a sub-score for a particular location.

A. Phase Zero of The Framework

1) *Definition of the problem: Gas must be paid with ETH for every transaction in Ethereum expansions. Therefore, in order for*

Hunters to act on behalf of such addresses, they must somehow feed their accounts with non-zero ETH transactions. This suggests that Sybil Hunters form ETH-transfer clusters between all of their addresses, and our studies show that these cluster patterns often resemble established Sybil patterns to some degree of noise. As seen in [8], Sybil ETH transfer clusters often take the shape of star, chain, or hybrid patterns. Furthermore, broad Sybil clusters discovered by

2) other detection solutions fit into such schemes, for example, as in [6], [7].

The plan is to use a blockchain community graph pattern recognition algorithm. It assesses if a cluster is more noisy and belongs to the "organic" normal class, which would be more likely to include poisoned honest addresses, or whether it is more akin to a flawless Sybil structure. It should be noted that a lot of Layer2 solutions employ their own versions of ERC-4337 [19], which makes it possible to pay fees via Paymasters. Like ERC-4337, ZKsync also uses native account abstraction. There are many different kinds of Paymasters, but the two primary ones are Sponsored (Gasless) Paymasters and ERC-20 Paymasters [20]. Since accounts must utilize ERC-20 tokens rather than ETH and those tokens are transferred to Sybil addresses via funding addresses, tracking the use of ERC-20 Paymasters is simple. Since Sybil cluster topologies would be same in both scenarios, the pattern recognition technique would function on valid ERC-20 token transfer-based clusters as it did with ETH. The other kind, Sponsored Paymasters, doesn't need the address that sent the transaction in order to have tokens. Sponsored Paymaster willingly pays for all petrol expenses at no cost to the user. The transaction receipt may be used to monitor Paymaster use, especially if the transaction is of the EIP-712 [21] type and the Paymaster field is set to a nonzero address [22]. In this case, Sybil cluster topologies could vary from those in ETH or ERC-20 funding transfers; in fact, they might not even match the topologies described in the literature. We assume that every Sybil Hunter we look for pays fees in ETH and uses more than just a Paymaster in order to streamline our Proof of Concept for phase zero and to comply with topologies outlined in references like [8]. These choices also expedite calculation.

1) Graph Isomorphism Network In relation to the graph isomorphism network that we used in our study, the publication [23] provides an excellent introduction to the subject of graph neural networks. We concentrate on the graph-level binary classification issue in phase zero. The objective is to accurately classify an unknown graph g using a model f, i.e., yg = f (g), yg 0, 1, by training the model in sets G and Y, given a collection of undirected graphs g1,..., gn G and a set of their labels y1,..., yn Y.

The graph isomorphism network (GIN) [24], a neural network that is a member of the class of Graph Neural Networks (GNN), has been selected as the inductive machine learning approach for this purpose. Graphs that are transformed into tensors with edge indices, node characteristics, and graph labels are fed into GNNs. In the training process, they enable us to represent graph topology. GIN updates the node embedding using the formula [24] for each iteration 0,..., k.

h(k) = MLP (k)((1 + ϵ(k)) N (v) is the 1-hop neighborhood of node v, hv is the embedding node, and MLP is a multilayer perceptron. A graph representation for a prediction in the graph-level classification challenge is provided by:

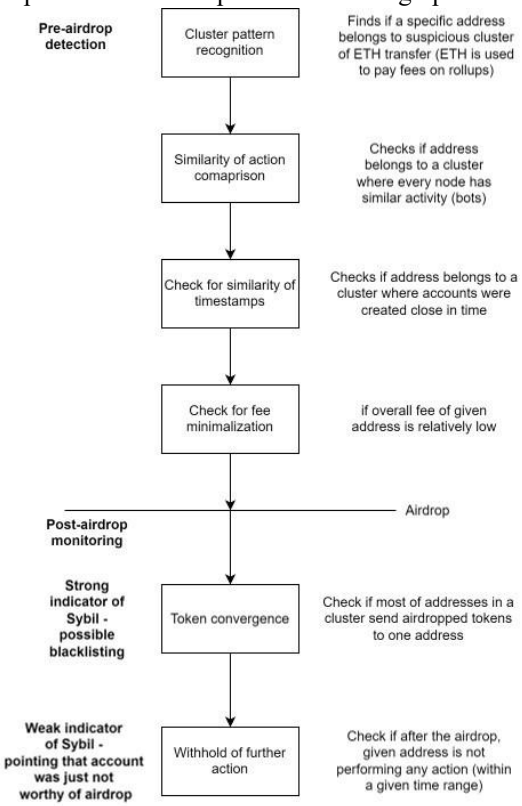


Fig. 1. Proposed Sybil detection framework structure

$$h_G = Concat(Readout(h^{(k)}|v \in G)|k = \{0, ..., K\}). \quad (2)$$

Readout is a global total in this case. K = 5 was our choice. The final prediction scalar is then obtained in our framework by performing linear transformations on hG using the RELU activation function. After applying a sigmoid function to a scalar, a binary classification is produced. GIN is a great choice for differentiating nonisomorphic graphs and hence a suitable tool for our goal since it is as expressive as the Weisfeiler-Lehman graph isomorphism test [24].

1) Data preparation and training: To start, there aren't many readily available labeled Sybil clusters at this time, which is definitely insufficient to train our model. Consequently, two methods have been tried:

•using artificial data to train the model,

By manually categorizing clusters and using a combination of fake and actual data to train the model.

Hypothesis: Manually searching and labeling blockchain graphs is more difficult than creating a decent synthetic dataset and training a model on it. We test a predetermined hypothesis for the first method. We can generate a lot of synthetic data because we know what Sybil communities often look like. For every subclass—stars, chains with cycles, and hybrids—500 graphs were created, each with a predetermined noise level—that is, random edge addition between random vertices—so not every graph is flawless. There were between 10 and 100 edges. Random graphs with $0.01 < p < 0.1$ for the usual class and scale-free graphs with lowered hub generation parameters, also with threshold noise were chosen. The number of normal graphs was almost the same as in the Sybil class. It was ensured that, in the entire generated dataset, all samples were unique.

The second method collected transactions with ETH transferred larger than 0 and without input data by dumping Ethereum Mainnet data between blocks 21,500,000 and 21,550,000. Blocks ranging from 1 to 1,500,000 were selected from the ZKsync data dump [25]. Additionally, ZKsync only recorded financing transfers between EOAs in order to eliminate noise from smart contract exchanges. Maximal linked components have been identified on such prepared transaction sets. For Ethereum and ZKsync, components with node counts ranging from 10 to 300 and 10 to 900, respectively, were labeled. One individual carried out the labeling procedure by hand. If a graph met all of the following criteria, it was deemed normal:

• There was no ideal star, chain, or hybrid topology in the graph.

• For most accounts, the amount of ETH moved between two specific accounts was sufficient to be deemed natural, meaning it was at least 0.01 ETH;

• A big chain structure, or more than ten addresses linked in a perfect chain (vertex degree of two) that was a part of the connected component, was not visible in graph topology.

•A star sub-topology with a large number of addresses might not be ideal; that is, there should be an edge between a small number of non-hub addresses.

If the graph matched a known Sybil topology, it was either skipped or labeled as Sybil. Additionally, human examination of address behavior was carried out for certain disputed graphs, those whose group assignment was uncertain, in order to look at previous activity and determine the probability of bot control. It produced 1014 actual Sybil graphs and 38 real normal graphs overall. 300 artificial normal graphs were created in order to lessen bias. Seventy-five more synthetic Sybil graphs were added to the data set in order to provide some almost ideal hybrid structures.components here just for simplification of showing the concept. Normally, on components with a number of addresses greater than y, a cluster detection algorithm should be run, e.g., Louvain community detection [26]. Such clusters, together with other MCCs, are then to be fed as input to phase 0.

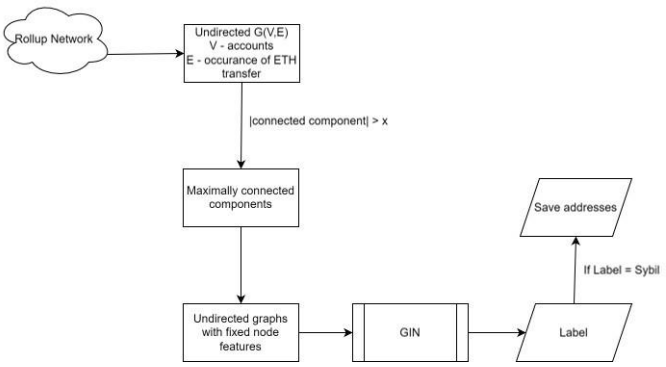


Fig. 2. Binary classification algorithm

B. Phase One of The Framework

Phase 1 is a follow-up to [8], and we slightly adjusted their methodology to suit our requirements. Smart contract interactions result in log entries with certain fields, referred to as topics. Each triggered event, such a swap or transfer, is identified by a distinct hash in Ethereum-based networks. Even a base token transfer across EOAs, like as Rollup's version of ETH, requires a system smart

contract on ZKsync. A sequence including topic0 hashes is generated for each account in MCC; however, if the same hash appears more than once, a dedicated iterator to that hash is concatenated with it. Next, sets of Cartesian products of address event sequences are constructed using ordered pairings without reversal and without pairs with identical elements. The Jaccard coefficient (JC) is then used to compare these products. Ultimately, a similarity matrix is constructed and put into the DBSCAN algorithm. The Jaccard coefficient for sets A and B may be computed as follows: There are no actual features since this is a topology-only characteristic Sybil cluster recognition/classification challenge; each node is given a number of 1. A stratified 5-fold has been used to divide the data into training, validation, and testing sets. Approximately 1027 samples are used for training, 115 for validation, and 285 for testing. Mini-batching has been used to train the model.*4)*

Suppose that we have two addresses with very similar se-quences, i.e.: $S_A = \{e_i | i \in 0, .., n-1\}$, $S_B = \{e_i | i \in 0, .., n- k - 1\}$, where $e_0 = hash(Event)$, $e_1 = hash(Event)|1$, $e_2 = hash(Event)|2$, etc. Assuming that A and B designate sets of Cartesian products on S_A and S_B, $|A| = n$, $|B| = n - k$, $B \subseteq A$. Then the Jaccard coefficient is calculated as:Figure 2 shows the flow of the phase 0 algorithm. First, an undirected graph of the entire network is created. Vertices are blockchain addresses and edges say if between two addresses an ETH transfer ever occurred. Then on such a graph, max-

connected components (MCC) are found. Then GIN classifies the MCCs that satisfy $|MCC| > x \wedge |MCC| < y$,

means that if the Sybil address performs fewer transactions than other Sybil, but they both maintain the same order of transactions, in an appropriately long event sequence, they should exhibit a high similarity coefficient, which is desirable and demonstrates that bots cannot easily deceive the algorithm by not creating k of the last events from sequences of some addresses. To have a lower JC score, addresses would have to switch order of events or just perform different events, and both are harder to automate.

DBSCAN [27] is a density-based clustering algorithm dedi-cated to noise-free data sets, taking two parameters: *mintPts* and ϵ. Based on [28], the recommended value for minimum points to form a cluster is $minPts = 2\ dim$. However, in our unique case, the dimension is the number of samples, so such an idea will not fit. We then try a different approach, where $minPts =\ \ dim$. There exist methods to determine the optimal parameter ϵ, such as in [28]–[30]. The popular method is to create a plot of sorted k-th nearest-neighbour distances from dataset samples and then find a knee on curve. However, a method presented in [8] uses a silhouette coefficient with grid search to estimate the best ϵ, and since vector coordinates are always between 0 and 1, we shall use this approach. We set the maximal ϵ in the grid search to 0.5, since our primary task is to detect Sybil similarities, not clusters per se.

Every address that is not marked by DBSCAN as a noise point is given a score of 1. However, it is not sufficient for all Sybil topologies, especially stars, where the hub account has significantly different log entry than its neighbours. Our solution for that is to find 1-hop neighbours of all clustered addresses and mark them. To balance judgment between real Sybils and poisoning attack victims, one can consider setting a lower score on such addresses. We decided to stick with a score of 1.

C. Phase Two and Three of The Framework

In phase two, accounts within MCC that were created close in time are being found. By creating an account, we consider the first transaction or event in which the "to" attribute or topic2 contains the address of this account. The corresponding block to that action is then found and from this block a timestamp is extracted. The timestamps are then fed to the DBSCAN algorithm in raw form. We assume that if two accounts were created within 4 hours, their creation time is systems are used. Moreover, our goal is to detect accounts that exhibit strong Sybil-like characteristics. Therefore, based on the expertise in the field, we chose ϵ to be 14400 in epoch time. The minimum samples to form a cluster have been chosen to be 3, since it is a one-dimensional case, and also only two accounts created within four hours could be coincidence. In addition, sparse clusters are not considered undesirable.

In phase three, the MCC account is checked if its average gas usage of all transactions is less than the average gas usage of all transactions of the entire network. Again, as in all previous phases, only EOAs are considered, and only the average gas usage from EOAs is chosen as the threshold. The distribution of transaction average gas usage recall gamma distribution, and it has been shown on the figure 3. When creating the distribution, we considered only addresses that had initiated at least one transaction. The mean μ has been calculated to be 627,361 units and the standard deviation σ is 347,242 units. All accounts with gas usage smaller than $\mu\ \ \sigma$ are marked.

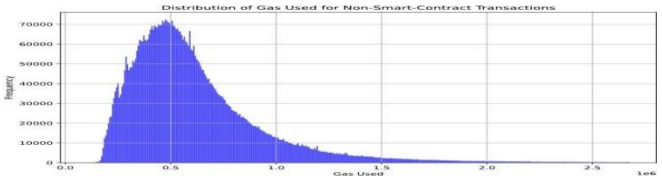


Fig. 3. Average gas usage distribution of EOAs

D. Merging scores

The last step is merging the results from all stages. We propose the following formula for calculating the final score of a given address:

considered similar, as a large gap between account creation times may lead to false positives. Based on how DBSCAN works, with our assumption, the two density-reachable border

$$Score = \frac{W_0 \cdot P_0 + W_1 \cdot P_1 + W_2 \cdot P_2 + W_3 \cdot P_3}{W_0 + W_1 + W_2 + W_3}$$

points in a cluster can be up to 4 hours apart. Therefore, even a cluster where accounts are created every 4 hours can still be identified as a single group. There is no universal time threshold as each cluster may have accounts created at different intervals. Finding a single optimal value based on all clusters could lead to false positives; that is, it would favour a distance chosen specifically to include as many accounts as possible, which would be incorrect. We consider 4 hours to be a short enough time window that, if multiple accounts within a cluster are created during this period, it suggests unnatural behaviour, such as the use of a script to generate accounts. Of course, attackers may choose to create accounts over a longer time span (e.g., one per day), but this increases the duration and cost of the attack, particularly if automated

$$s.t.\ P_i \in \{0, 1\}, W_i \in (0, 1] \qquad (7)$$

W_i in formula 6 represents the weights chosen for a specific phase. P_i are the results of a given phase of the framework. Each address in the data set can have a different value P_i, and each result is obtained by applying the algorithm of the i-th phase. To find the best weights, a grid search is performed. To find the optimal set of weights, an F1 score was used. We will manually label addresses from the created MCC pool. The labels will be assigned according to the post-airdrop monitoring part in Figure 1 and by looking at the general characteristics of the accounts and the clusters they are in. Errors of the first and second types will be determined using

a fixed threshold th, i.e., $Score \geq th \Rightarrow \{0, 1\}$. The best weights will be determined for $th \in \{0.1, 0.2, ..., 0.9\}$.

III. Results and Their Analysis

A. Phase Zero

On mixed real and synthetic data, model hyperparameters,

i.e. number of neurones per layer, batch size, epochs, dropout, and learning rate, have been fine-tuned using a grid search. With a learning rate equal to 0.00005, the ADAM optimisation algorithm has a satisfactory loss descent speed and improves throughout the epochs. The batch size of 64 has fewer sudden spikes on both the accuracy and loss validation curves than the batch size of 32 samples. The loss accuracy validation plot for the best configuration is shown in Figure 4. Evaluation metrics were calculated that included BCELoss, accuracy, F1 score, and balanced accuracy. The best results were obtained for the model trained with 100 epochs, 32 neurones, and the first fold:

- Loss: 0.00199,
- Accuracy: 0.98252,
- F1 Score: 0.98861,
- Balanced Accuracy: 0.96830.

This model was trained without dropout. For dropout with probability of $p = 0.4$ some perfect stars were incorrectly labelled, changing it to $p = 0.2$ removed this problem; however, it still resulted in lower accuracy than with $p = 0$. Moreover, using positive dropout resulted in less stable loss validation curve, hence our decision.

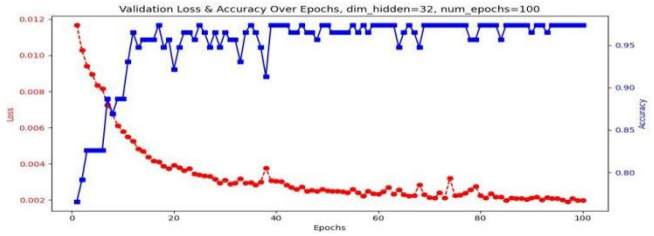


Fig. 4. Loss and accuracy over epochs in validation step

For ZKsync roll-up blocks between 1 and 29,710,000, produced before snapshot, were analysed. It gave a total of 1565 MCCs with a number of vertices ranging between 10 and 900. From that, 1529 were labelled as Sybil topologies, which gives 29,889 blockchain addresses. Figures 5, 6, and 7 show examples of labelled ZKsync topologies.

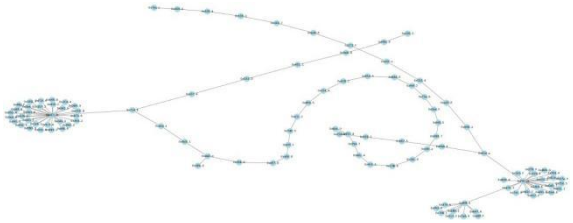


Fig. 5. ZKsync Sybil complex star - chain hybrid cluster

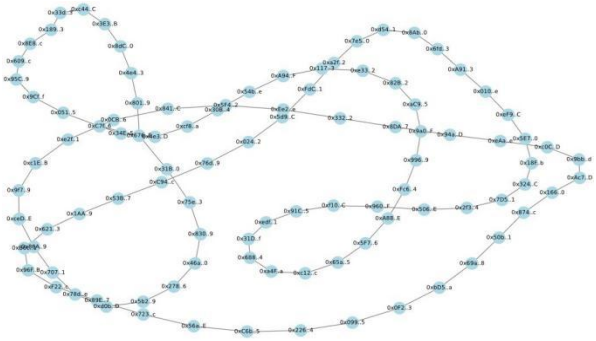


Fig. 6. ZKsync Sybil chain / cycle cluster

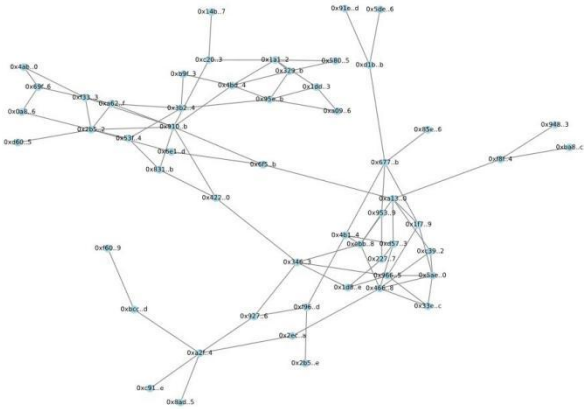


Fig. 7. ZKsync topologically normal cluster

For the approach using only synthetic data for training, although the accuracy of validation and the accuracy of the test (synthetic) were very high >98%, the model was actually useless. We tried to classify real ZKsync-connected components with such trained GIN and found that it has trouble with classification of „obvious" Sybil topologies, even including perfect star structures. Accuracy on real dataset was also computed which turned out poor and it did not matter how big the training dataset was or how well tuned hyperparameters were. It would suggest that model tends to extremely overfit on synthetic data to the point where it looks more like it remembers structures instead of generalising them (perfect star example). We also tried generating graph with different noise levels between vertices, and the results were always poor. In addition, test evaluation metrics were computed on the real sample dataset:

- Loss: 1.07751,
- Accuracy: 0.07415,

- F1 Score: 0.07590,
- Balanced Accuracy: 0.51972.

It could be possible that, although we applied some random-ness in generating samples, instances within a class were too similar to each other for GIN model, even if visually they appeared to be distinct enough. Additionally, samples from normal and Sybil sets may have been too distinct from each other, making it difficult for the neural network to generalise effectively. For such reasons, we deny the hypothesis from chapter II-A.

B. Other Phases

The number of MCCs, when an edge between EOAs repre-sents an arbitrary transaction, was 1645. In all those connected components, there were 35,843 addresses.

1) Phase One: A total of 34,636 addresses form clusters that were marked by DBSCAN. Its a very high percentage of the dataset, but the probable reason for that is the same as the finding described in results of phase three. From that, 2592 addresses were marked as 1-hop neighbours of those Sybil accounts.

2) Phase Two: In this algorithm, 22,682 addresses were located in some cluster found by DBSCAN. We observed that many connected components were of perfect star topology, where a hub address sent one transaction to each of its neighbours, very close in time.

3) Phase Three: A total of 26,493 addresses have been marked in phase 3. However, it has been observed that most of them, 24,651 have gas usage of 0, meaning they did not create any transaction. Also, usually they have really small balances, and, of course, such addresses would never be eligible. Such accounts have been created by hubs, which can be users who wanted to generate synthetic liquidity in order to be eligible. With such a large scale, it is not excluded that some users might have followed the same tutorial. It is up to the project team whether or not they desire to ban such hub addresses.

C. Final Results

We manually labelled 13 normal and 85 Sybil addresses. Let MCC_0 denote all components maximally connected from phase 0 and MCC_1 from other phases. For the labelling process, addresses a were selected that meet:

$$\{a : (a \in MCC_0 \cap MCC_1) \wedge (\sum_{i=0} P_{ia} > 0)\}. \quad (8)$$

Sybil score thresholds ≥ 0.2 gave the same, best F1 score of 0.9492. We chose a threshold of 0.6 as the definitive boundary to ban addresses or not. For such a threshold, the best weights were selected, although there were many others to choose from. The choice was also based on the fact that phases 0, 1 and 2 indicate a Sybil behaviour more than phase 3. The following weights were selected: $W_0 = 0.9$, $W_1 = 0.5$, $W_2 = 0.6$, $W_3 = 0.3$. In table I the results of our framework are presented, that is, the final calculated Sybil scores for all addresses analysed.

More than 37,563 addresses exceed the safe threshold and can be confidently marked as Sybils. To recall, the total number of EOAs analysed is 49,249, so around 76% of all accounts resemble a strong Sybil behaviour with respect to the proposed framework. That is a high percentage of the overall population, however, it can be explainable. As a result of Phase three, a total of 24,651 have not made any transaction and in many

TABLE I
NUMBER OF ADDRESSES PER SYBIL SCORE

Score	No. of addresses	Percentage share
0.0000	161	0.3%
0.1304	20	0.9%
0.2174	293	
0.2609	123	
0.3478	8027	22.5%
0.3913	1831	
0.4782	1149	
0.5217	82	
0.6087	3059	71.0%
0.6522	1485	
0.7391	14375	
0.7826	94	
0.8696	15936	
1.0000	2614	5.3%

cases did not transfer any assets to other addresses. Therefore, those accounts must have been a part of perfect star topologies with one very active hub account, possibly the only contender for the airdrop in a given topology. The hub's use of „dead" accounts to simulate activity through small ETH transfers supports the conclusion that the large number of accounts marked as Sybil is not a misclassification.

For further analysis, we compare our results with the ZKsync eligibility list. This list can be found at https://github.com/ZKsync-association/zknation-data and con-tains a list of eligible blockchain addresses, including EOAs. ZKsync did its own Sybil detection, and a brief explanation of it can be found in [10]. We checked common addresses between our Sybil list and their eligible addresses, and found that 342 addresses, with a Sybil score grater or equal to 0.6, can be found in the ZKsync list. Those common accounts received a total of 1,571,708 ZK airdropped tokens, and based on the market value at the time, the value of the rewards ranged between $270,000 and $300,000. When comparing the scale, there are not many entities, possibly because those are only EOAs from restricted MCCs. To have a full comparison, one would have to run our framework on clusters with unlimited size, including EOAs and smart accounts.

Moreover, we can notice that from 37,563 addresses labelled as Sybil by our framework, 37,221 of them were not present in ZKsync list meaning either they were simply not eligible or were excluded by ZKsync's Sybil detection system. This further indicates that the result of our framework did not have a high false-positive rate.

IV. Discussion

Why use GIN and not a non-machine learning pattern search algorithm? Because GIN output is not easily predictable and interpretable. As we can see in result there are clusters labelled as normal, however, in them there are also some Sybil-like patterns (false negatives) and (false positives) in normal instances. So Hunters cannot be sure that the topology they have created could fool the model. However, the black-box

nature of neural network can also be seen as a disadvantage as airdrop issuers would be interested in knowing exactly why given addresses were marked by the model, to provide transparency for project community.

To increase performance of the model, one's goal would be to require more labelled data. However, we cannot ever be sure if a given address is a Hunter without self-reporting. Even if an address belongs to, for example, a perfect star cluster, it's chances of being a Hunter are higher but still, we are not sure. That is one of the reasons why it is hard to create a correct large dataset of real-life Sybil graph cases. The aftermath of detecting Sybil using a topology-based approach can be, naturally, the evolution of Sybil cluster structures. The hypothesis would be that the more clusters one detects, the more Hunters try to fool the detection algorithm, and therefore, gradually create more noisy graphs to the point where Sybil clusters would look like graphs from a normal class. It could completely negate the point of transfer topology-only detection, although, it would also make Sybil attacks on airdrops more expensive.

A big problem in Sybil identification in rollups is that not every address operated by a human is an EOA. There are also smart accounts present. However, creation of EOA does not require any transaction, so it is free in contrast to smart account, where for one to be deployed a special transaction must be sent through interaction with a system smart contract. For that reason, EOAs are more likely to be used in a Sybil attack scheme, and that is one of the reasons why we decided to not bother identifying smart accounts. Teams that try to detect Sybil clusters could make a large effort to identify which addresses are from ERC-4337 and which are smart utility contracts. Furthermore, an account should not be marked as Sybil because it sent a transaction to a smart contract, for example, a Swap, that has a very high degree. Also, DEX and CEX addresses should be searched and excluded.

Referring to the analysis of phase 3 in Section III-B3 of our framework, many EOAs had not created any transaction. There is an idea to add one more phase to our framework, which would be based on marking 1-hop neighbours of accounts with average gas usage of 0, if in a given clusters there are many such „dead" addresses. From our observation, such 1-hop neighbours usually could be hubs in the local star topology. Moreover, we distinguish two types of hunter attacks. In the first, hunters create complex clusters in which there are directly or indirectly connected addresses that seek to receive tokens from the airdrop. In the second approach, hunters can create many isolated clusters inside which only one account (hub) attempts to become eligible, and all its neighbours are used only for interaction with that specific address.

Referring to phase 2 in Section II-C, an improvement can be applied that potentially increases the detection accuracy with the cost of additional computation. The parameter ϵ of DBSCAN could be dynamic in the sense that for a given set of addresses, a DBSCAN is run many times with different ϵ, e.g.: $\epsilon \in 1h, 4h, 8h, 16h, 24h,$ Accounts belonging to clusters achieved with more ϵ instances have a higher score for the phase 2 algorithm. For example, if accounts are in cluster for time windows of 24, 16 and 8 hours, their score would be higher than those that only were in cluster for a window of 24 hours. In addition, the phase score would no longer be binary, i.e. $P_2 \in [0, 1]$. Such an approach could strike a balance between minimising false positives and avoiding the challenge of determining an appropriate time interval between account creations.

V. Conclusion and Future Work

Numerous possible Sybil ZKsync accounts have been identified in this article. In actuality, the suggested algorithm chain functions as a sieve, allowing addresses with a lower score to be further examined using various techniques while those with a higher score may be promptly eliminated from airdrop. It should be mentioned that too stringent wallet filtering—that is, labeling too many as Sybil (false positive)—may deter helpful, sincere people from participating in the initiative. The most notable Sybil players may be captured using our methodology. We can be more versatile when it comes to the false positive rate since the score threshold for designating an address as malicious may be determined freely.

The airdrop's design has a significant role. Airdrops that distribute the whole fixed pool of tokens across eligible addresses are one kind of airdrop, while others have a predetermined token quantity per user. This would imply that each address would earn less tokens the more eligible users—including Hunters—are, making the Sybil assault less economical. To reduce competition, a poisoning attack would be more likely under such a setup.

Additionally, it should be noted that there may be online guides explaining how to participate in an airdrop, causing several addresses to carry out comparable tasks on the blockchain. However, unless their addresses are extremely near to one another in an address-transactions topology or if they are part of the same cluster, honest users may happily follow the same tutorial since our system is also somewhat immune to such situations. Next, there are websites that facilitate the search for potential inbound airdrops [31]–[33], which may assist Hunters in selecting the network to carry out the Sybil assault. When their project is advertised on such a website, developers who want an airdrop should exercise extra caution.

In terms of next projects, a fresh method for GIN training may be created. One would be motivated to train GIN on directed graphs if additional Sybil clusters are identified, since this might better capture the characteristics of suspicious clusters. We believe that star topologies with primary Sybil accounts, or hubs, often feature only outgoing directed edges from such central accounts, but not incoming edges, based on our findings made during the framework's development. Using account-level characteristics to train the machine learning model and perhaps including it as an additional stage in our approach is another suggestion. In this case, a model would identify possible Sybil accounts according to their behavior and statistical characteristics, such as degree or measurements of a graph vertex's importance. The defining of Sybil address properties may benefit greatly from such a method.

Running our approach on datasets containing smart accounts in addition to EOAs would be a substantial effort. In-depth

A fantastic framework for blockchain data preparation may be created by demonstrating the procedures for building such a dataset, such as community discovery and contract screening.

Additionally, the suggested methodology may be compared to other Sybil detection approaches, perhaps using a small tagged or synthetic data set. The features of the outcomes for each framework, together with the overall accuracy and comparable measures, might be shown in such a comparison. It would enable projects to choose the best strategy.

REFERENCES

[1] [1] The Block, "We made close to $1 million inside the murky world of airdrop farming," accessed: 2025-02-09. [Online]. accessible: https://www.theblock.co/post/225215/we-made-close-to-1-million-inside-the-murky-world-of-airdrop-farming
[2] [2]Binance Square, "Magic eden airdrop frenzy: One farmer cashes in over $2 million," retrieved on February 9, 2025. [Online]. Accessed on February 9, 2025, at https://www.binance.com/en/square/post/17443215929586 [3]X. [Online]. Accessible: https://x.com/lookonchain/status/1639165562663358466 [4]J. Messias, A. Yaish, and B. Livshits, "Airdrops: Giving money away is harder than it seems," arXiv preprint arXiv:2312.02752, 2023.
[3] [5]A. Yaish and B. Livshits, "Tierdrop: Harnessing airdrop farmers for user growth," arXiv preprint arXiv:2407.01176, 2024.
[4] [6]TrustaLabs, "Airdrop-sybil-identification," accessed: March 13, 2025. [Online]. accessible at https://github.com/TrustaLabs/Airdrop-Sybil-Identification
[5] [7] Arbitrum Foundation, "Arbitrum sybil detection," retrieved March 13, 2025. [Online]. accessible at https://github.com/ArbitrumFoundation/sybil-detection [8] Z. Liu and H. Zhu, "Fighting sybils in airdrops," arXiv preprint arXiv:2209.04603, 2022.
[6] [9] C. Sguanci, R. Spatafora, and A. M. Vergani, "Layer 2 blockchain scaling: A survey," arXiv preprint arXiv:2107.10881, 2021.
[7] [10]ZK NATION, "Zk airdrop," accessed: April 13, 2025. [Online]. accessible at https://docs.zknation.io/zk-token/zk-airdrop/ [11]K. Lommers, C. Makridis, and L. Verboven, "Designing airdrops," SSRN 4427295, 2023.
[8] [12] D. W. Allen, "Crypto airdrops: An evolutionary approach," Journal of Evolutionary Economics, vol. 34, no. 4, pp. 849–872, 2024.
[9] [13]"Artemis: Detecting airdrop hunters in nft markets with a graph learning system," C. Zhou, H. Chen, H. Wu, J. Zhang, and W. Cai, Proceedings of the ACM Web Conference 2024, 2024, pp. 1824–1834.
[10] [14] Y. Qin, T. Ma, H. Chen, and H. Duan, "Artemix: A community-boosting-based framework for air-drop hunter detection in the web3 community," Blockchain, vol. 2, no. 2, pp. 1–24, 2024.
[11] [15]
[12] "Toward resilient airdrop mechanisms: Empirical measurement of hunter profits and airdrop game theory modeling," arXiv preprint arXiv:2503.14316, 2025, J. Luo, H. Kang, S. Zheng, and X. Liu.
[13] [16] S. Fan, T. Min, X. Wu, and W. Cai, "Altruistic and profit-oriented: Making sense of roles in web3 community from airdrop perspective," Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems, 2023, pp. 1–16.
[14] [17] "Threat analysis of poisoning attack against ethereum blockchain," T. Sato, M. Imamura, and K. Omote, IFIP International Conference on Information Security Theory and Practice. 2019, Springer, pp. 139–154.
[15] [18] "Blockchain address poisoning," arXiv preprint arXiv:2501.16681, 2025, T. Tsuchiya, J.-D. Dong, K. Soska, and N. Christin.
[16] [19] Ethereum, "Erc-4337: Account abstraction using alt mempool," retrieved April 13, 2025. [Online]. accessible: https://eips.ethereum.org/EIPS/eip-4337 [20]Zksync, accessed: April 13, 2025. [Online]. Accessible: https://docs.ZKsync.io/ZKsync-era/guides/ZKsync-101/paymaster [21] Ethereum, "Eip-712: Typed structured data hashing and signing," accessed: April 13, 2025. [Online]. accessible: https://eips.ethereum.org/EIPS/eip-4337 [22]Zksync, accessed: April 13, 2025. [Online]. [23] I. R. Ward, J. Joyner, C. Lickfold, Y. Guo, and M. Bennamoun, "A practical tutorial on graph neural networks," ACM computing surveys (CSUR), vol. 54, no. 10s, pp. 1–35, 2022; accessible at https://docs.ZKsync.io/ZKsync-protocol/account-abstraction/paymasters.
[17] [24]K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" arXiv preprint arXiv:1810.00826, 2018.

[18] [25] "A public dataset for the zksync rollup," arXiv preprint arXiv:2407.18699, 2024, M. I. Silva, J. Messias, and B. Livshits.
[19] [26]V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," Journal of statistical mechanics: theory and experiment, vol. 2008, no. 10, p. P10008, 2008.
[20] [27]M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al., "A density-based algorithm for finding clusters in large spatial databases with noise," in kdd, vol. 96, no. 34, 1996, pp. 226–231.
[21] [28]E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "Dbscan revisited, revisited: why and how you should (still) use dbscan," ACM Transactions on Database Systems (TODS), vol. 42, no. 3, pp. 1–21, 2017.
[22] [29]A. Starczewski, P. Goetzen, and M. J. Er, "A new method for automatically determining the dbscan parameters," Journal of Artificial Intelligence and Soft Computing Research, vol. 10, 2020.
[23] [30]Autoepsdbscan: Dbscan with eps auto-matic for huge dataset, M. N. Gaonkar and K. Sawant, International Journal on Advanced Computer Theory and Engineering, vol. 2, no. 2, pp. 11–16, 2013.
[24] [31]Alpha Drops, accessed: April 6, 2025. [Online]. accessible at https:
[25] //www.alphadrops.net/ [32]Bankless, accessed: April 6, 2025. [Online]. Accessible: https://www.bankless.com/airdrop-hunter [33]Airdrops.io, retrieved on April 6, 2025. [Online]. Accessible: https://airdrops.io/