INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2026, VOL. 72, NO. 1, PP. 1–9
Manuscript received September 16, 2025; revised February, 2026. doi: 10.24425/ijet.2026.157879

Airdrop Sybil Attack detection framework supported by machine learning

Kamil Kaczyński and Aleksander Wiącek

Abstract*—Airdrop Sybil attacks can be a lucrative labour, and tokens received from one airdrop by an effective hunter can reach thousands of dollars. Sybil attacks in this context are not always desired by projects and are often seen by honest players as inappropriate behaviour, which can reflect badly on a project's reputation. For such a reason, it is well expected that Sybil attacks detection systems will be constantly improved. In this work, a multistep framework is presented. Its idea is to sort blockchain addresses and assign them a score that will indicate if a given address is closer to a normal or a Sybil class. A graph isomorphism network was used to classify topologies, and its parameters were tuned on a dataset labelled by the authors. In other steps, a DBSCAN was used for the account clustering task. Users of the framework can assign arbitrary weights to each step, which will determine how important a step is to them and result in a different score for a given address. The best weights were found with a grid search method as well as a threshold after which the address is considered Sybil. In this paper a set of EOAs from ZKsync rollup was analyzed. In the end, 76% of all the accounts analyzed were marked as Sybils. Compared to the official ZKsync eligibility list, we found 342 addresses that received airdrop tokens but were marked as Sybil by our solution.

***Keywords*—Blockchain; Airdrop; ZKsync; Sybil Attack Detection; Graph Isomorphism Network**

I. Introduction

An blockchain airdrop is a strategy used by project creators to attract user attention and increase on-chain activity. It typically involves offering rewards for completing specific tasks related to the project. Airdrop creators usually commit to distributing (airdropping) reward tokens to eligible addresses. This is a common practice among newly launched projects, but developers can also create several airdrop events throughout the project lifetime, for example, as in the Optimism rollup. Airdrops can have a unique design, and there is a visible split between proactive and retroactive ones. In purely retroactive design, users receive tokens based on their past behaviour, and the list of instructions to be eligible is shown after the snapshot of the network is taken. In contrast, proactive design clearly indicates what actions one must take to receive tokens. Both are prone to so-called airdrop farmers to some extent.

K. Kaczyński is with Faculty of Cybernetics, Military University of Technology, Warsaw, Poland (e-mail: kamil.kaczynski@wat.edu.pl).
A. Wiącek is with Faculty of Cybernetics, Military University of Technology, Warsaw, Poland (e-mail: aleksander.wiacek@student.wat.edu.pl).

Airdrop Sybil attack or airdrop farming is a type of attack where a single / few entities known as farmers or hunters create many fake accounts in order to outsmart the airdrop and greatly increase the pool of tokens that they receive. Sometimes, the hunters' value of assets from an airdrop can be extremely high, reaching several thousands of dollars [1]–[3]. In 2023, a hunter managed to collect 933,365 Arbitrum tokens with a total value of more than 1 million dollars [3]. This shows the scale of Sybil attacks, as it was one of many similar cases across the entire cryptocurrency industry.

Airdrop farming is widely regarded as something negative and not fair to honest users. Hunters usually rapidly sell tokens after receiving them [4], which impacts the market and could force honest token receivers to do the same. Also their successful presence might discourage the overall public relation to the project. Hunters generate synthetic network activity with the sole purpose of simulating engagement to profit from incentives, rather than contributing genuine value. Developers try to mitigate Sybil attacks by a careful airdrop designs including application of proof-of-humanity, address blacklisting, self-reporting, or reward design that is not easily spendable. However, some can argue that airdrop hunters are not entirely evil and can attract some newcomers [5]. The project should consider in particular what tactics should be adopted to deal with the Sybil attack phenomenon.

Simultaneously, another area of attack prevention is the detection of Sybil accounts, and so this topic is further explored within this paper. Sybil attacks may be detected by investigating clusters in a graph created based on transactions between accounts. Moreover, some structural / transactional patterns associated with Sybil attack have been recognised [6]–[8]. In addition, it is more common to see the usage of machine learning models to execute detection tasks. On the other hand, hunters adapt to avoid detection [1]. Sybil attacks may be performed by organised groups, containing many diligent people, that could be hired with one goal - to game airdrop and split the reward. A worker could manually and consistently create, fund, and operate many accounts, excluding the use of bots, and implement complex transactional patterns between blockchain addresses. Naturally, such attacks would overall cost more to perform than the attack in which a single hunter uses bots.

Rollup is a Layer 2 solution whose main purpose is to scale the Ethereum network. Rollup is a blockchain that can handle a significantly higher volume of transactions per second.

Transactions are processed on the Layer 2 network, while Layer 1 (Ethereum) only receives proofs of state changes. Rollups use a smart contract on Layer 1 that tracks the Layer 2 state via a Merkle root [9].

One of the rollups based on zero-knowledge proofs is the ZKsync Era network, and such a blockchain was the target of our analysis. Transactions from ZKsync blocks are compressed and submitted to Ethereum in a single batch. Therefore, ZKsync fees are much lower than on Ethereum. This efficiency has made it an attractive option for users and opened up a new opportunity for hunters. Lower transaction costs surely encourage Sybil behaviour. ZKsync team chose a retrospective airdrop model, evaluating user activity from the network launch on February 14, 2023, to March 24, 2024. A total of 17.5% of the entire supply of ZKsync tokens was allocated for the drop [10]. In March 2024, hundreds of thousands of wallets became eligible for a ZKsync airdrop. The tokens were distributed between two main groups: network users and contributors such as developers and researchers, each with their own set of eligibility rules. For users, meeting at least one of seven criteria qualified an address; examples include interacting with ten or more non-token contracts, swapping ten different tokens or using Paymasters at least five times. The amount of tokens received also depended on how early and how much in assets an address bridged to ZKsync. Additional multipliers were applied to accounts that showed signs of genuine human activity or a valuable contribution.

A. Paper Organization

In Section II the methodology of the proposed framework is presented. The choice of machine learning algorithms, their brief explanation with important parameters, has been included. In Section III the results obtained have been shown and suggestions of what they are likely to come from. The chapter includes neural network fine-tuning findings and also results of applying the framework to the ZKsync rollup dataset. Section IV presents a discussion of results and topics related to the framework and area of research that the authors find important. The last section V contains the conclusion and future work.

B. Related Work

The topics covered in the work are relatively new, so the literature directly related to the detection of Sybil attacks on blockchain might be limited. In [11] general strategies for airdrop design and other aspects are shown, such as Sybil attacks or implementation methods. This work can give a great overview of the research area related to airdrops. Another paper on different airdrop designs and a description of real-world examples is [12].

In [8] the authors propose a Sybil detection solution based on finding clusters of accounts with similar event history with the addition of searching for some known Sybil-like topologies. Their analysis was done on Hop protocol which supports Layer 2 solutions. Examples of hunter behaviour in the network were also shown, including how sibilant graph topologies are formed. The whole paper is a great addition to the whole area of airdrop Sybil detection.

In papers [13] and [14] a detection system for Sybil attacks in transactions with non-fungible tokens is presented. They called it ARTEMIS and it was supported by a neural network. Moreover, the authors provided some comparison of metrics between their composed neural network and some other machine learning methods, including popular graph neural networks such as GIN. In addition, a view of typical hunter trading patterns was included.

The authors in [15] propose a game theory model that simulates the dynamics between hunters and projects. They Show available strategies and propose effective incentive models that improve detection while keeping operational expenses low. The research was also based on the analysis of Hop Protocol and LayerZero.

In [16], using ParaSwap as a case study, the authors introduce a data-driven approach to propose a role-based taxonomy. It helps identify the behavior patterns of the community participants. The paper also addresses the subject of airdrop hunters and can be a good complement to the general concept of airdrop Sybil attack addressed in our work.

II. Methodology

We propose a multi-phase framework supported by machine learning methods, as shown in figure 1. Due to the poisoning attack that hunters can carry out, only a topology-based approach may lead to false positives. In the blockchain area, different poisoning attack definitions can be found depending on the context, e.g., [17] related to malicious content posted on the network and [18] related to dust, zero, or fake contract transfers. In [18], phishing transfers are sent from an attacker whose address is synthetically generated to be as similar as possible to that of the victim. In our context, addresses can be completely different. A Hunter would send a transaction with very little value to unknown accounts to confuse detection systems by tapping into a different topology or, in an extreme scenario, to harm others, meaning that honest users would be marked as Sybils. Therefore, the framework tries to solve the problem of identifying most Sybils with as few misinterpreted honest users by analyzing various account statistics. Its idea is to calculate the Sybil score for all filtered blockchain addresses in a given network lifetime. The score is calculated by performing 4 algorithms, each one returning a sub-score for a given address.

A. Phase Zero of The Framework

1) Problem definition: In Ethereum expansions, gas must be paid with ETH for each transaction. Therefore, Hunters have to somehow feed their accounts with non-zero ETH transfers, to be able to perform some action on behalf of those addresses. This implies that Sybil Hunters create ETH-transfer clusters between all their addresses, and based on our observations, often those cluster patterns are similar to known Sybil patterns to some noise level extent. Typically, Sybil ETH transfer clusters form star, chain, or hybrid patterns, as shown in [8]. In addition, general Sybil clusters found by other detection solutions fit into such schemes, for example, as in [6], [7].

The idea would be to implement a pattern recognition algorithm of blockchain community graphs. It determines if a given cluster is more similar to a perfect Sybil structure or is more noisy and of the „organic" normal class, which would have a higher chance of containing poisoned honest addresses.

Note that many Layer2 solutions implement their versions of ERC-4337 [19], thus creating an opportunity to use Paymasters to pay fees. ZKsync also implements native account abstraction similar to ERC-4337. There are a few Paymaster types; however, the two main are ERC-20 Paymasters and Sponsored (Gasless) Paymasters [20]. Tracking the usage of ERC-20 Paymasters is easy, as accounts instead of ETH must use ERC-20 tokens, and those are transferred to Sybil addresses through funding addresses. The pattern recognition algorithm would run on legitimate ERC-20 token transfer-based clusters just as it was with ETH, since Sybil cluster topologies would be the same in both cases. The other type, the Sponsored Paymasters, does not require an address which sent a transaction to have any tokens. Sponsored Paymaster voluntarily covers the entire gas cost and at no expense to the user. To track the usage of Paymaster one can obtain information about it from the transaction receipt, more precisely if it is an EIP-712 [21] type transaction with the Paymaster field set to a nonzero address [22]. Here, Sybil cluster topologies could be different from those found in ETH or ERC-20 funding transfers, and, in fact, they could not resemble the topologies found in the existing literature. To simplify our Proof of Concept of phase zero and to be compliant with topologies described in references such as [8], we assume that every Sybil Hunter we try to find pays fees in ETH and does not only use a Paymaster. In addition, such decisions speed up computation time.

2) Graph Isomorphism Network: The paper [23] is a great introduction to the topic of graph neural networks, which is related to the graph isomorphism network that we used in our work. In phase zero, we focus on the graph-level binary classification problem. Having a set of undirected graphs $\{g_1, ..., g_n\} \in G$ and a set of their labels $\{y_1, ..., y_n\} \in Y$ the goal is to correctly classify a given unknown graph g with a model f, i.e., $y_g = f(g), y_g \in \{0, 1\}$, by training the model in sets G and Y.

The inductive machine learning algorithm that has been chosen for this task is the graph isomorphism network (GIN) [24], that is, a neural network that belongs to the class of Graph Neural Networks (GNN). GNNs take as input graphs that are converted into tensors consisting of edge indices, node features, and graph labels. They allow us to express graph topology in the training process. GIN, with each iteration $\{0, ..., k\}$, updates the node embedding according to the formula [24]:

$$h_v^{(k)} = MLP^{(k)}((1 + \epsilon^{(k)}) \cdot h_v^{(k-1)} + \sum_{u \in N(v)} h_u^{(k-1)}) \quad (1)$$

where h_v is the embedding node, MLP is a multilayer perceptron and $N(v)$ is the 1-hop neighbourhood of node v. For the graph-level classification task, a graph representation for a prediction is given by:

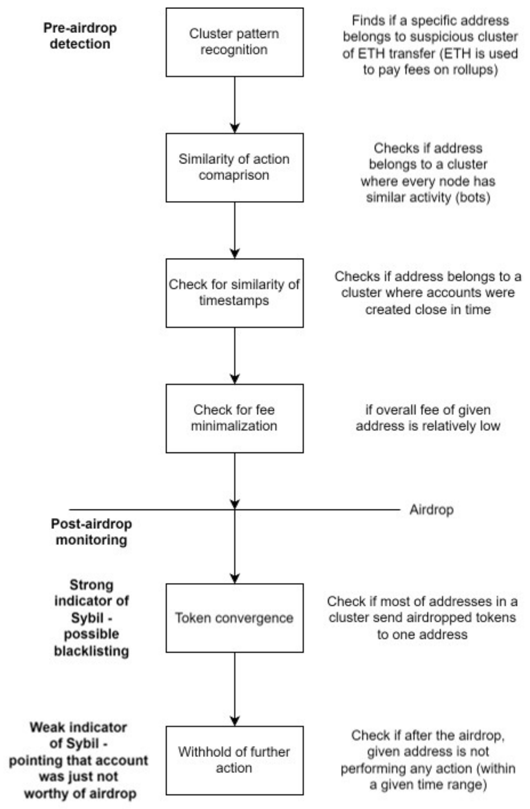
Pre-airdrop detection
Cluster pattern recognition
Finds if a specific address belongs to suspicious cluster of ETH transfer (ETH is used to pay fees on rollups)
Similarity of action comaprison
Checks if address belongs to a cluster where every node has similar activity (bots)
Check for similarity of timestamps
Checks if address belongs to a cluster where accounts were created close in time
Check for fee minimalization
if overall fee of given address is relatively low
Airdrop
Post-airdrop monitoring
Strong indicator of Sybil - possible blacklisting
Token convergence
Check if most of addresses in a cluster send airdropped tokens to one address
Weak indicator of Sybil - pointing that account was just not worthy of airdrop
Withhold of further action
Check if after the airdrop, given address is not performing any action (within a given time range)

Fig. 1. Proposed Sybil detection framework structure

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

Here, $Readout$ is a global sum. We chose $K = 5$. Then in our framework, on h_G we perform linear transformations with the RELU activation function to obtain the final prediction scalar. A binary classification is obtained after treating a scalar with a sigmoid function. GIN is as expressive as the Weisfeiler-Lehman graph isomorphism test [24] so it is an excellent option for distinguishing nonisomorphic graphs and therefore a promising tool for our task.

3) Data preparation and training: First of all, so far there are not many easily accessible labelled Sybil clusters, surely not enough to train our model. Thus, two approaches have been tried:

- training model on synthetic data,
- labeling clusters manually, and training model on mixed real data with synthetic data.

Hypothesis: *It is easier to generate a good synthetic dataset and train a model on it than to manually search and label blockchain graphs.* For the first approach, we test a given hypothesis. Because it is known what Sybil communities usually look like, we can create a lot of synthetic data. There were generated 500 graphs for each subclass: stars, chains including cycles and hybrids, all with a set noise level, i.e. random edge addition between random vertices, so not every graph is a perfect one. The number of edges ranged from 10 to 100. For the normal class, random graphs with $0.01 < p < 0.1$

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.

For the second approach, Ethereum Mainnet data were dumped between blocks 21,500,000 and 21,550,000, collecting transactions with ETH transferred greater than 0 and without input data. From the ZKsync data dump [25], blocks between 1 and 1,500,000 were chosen. Also, in ZKsync only transactions between EOA's were chosen to capture only funding transfers without noise associated with smart contract interactions. On such prepared transaction sets, maximal connected components have been found. Components with nodes numbering between 10 and 300 for Ethereum and between 10 and 900 for ZKsync were labelled. The labelling process was performed manually by a single person. A given graph was considered normal if it achieved all of the following requirements:

- the graph did not resemble perfect star, chain or hybrid topology,
- the ETH transferred between two given accounts for majority of accounts was high enough to be considered natural, i.e. at least 0.01 ETH;
- in graph topology there could not be observed a large chain structure, i.e. more than 10 addresses connected in a perfect chain (vertex degree of 2) that was a part of the connected component,
- if star sub-topology with many addresses was present, it could not be perfect, i.e. between few non-hub addresses there should be an edge.

Otherwise, the graph was skipped or labelled as Sybil if it resembled a known Sybil topology. Additionally, for some contested graphs, those whose group assignment was uncertain, manual analysis of address behaviour was conducted to examine past activity and assess the likelihood of bot control. In total, it gave 38 real normal graphs and 1014 real Sybil graphs. To reduce bias, 300 synthetic normal graphs were generated. To introduce some near-perfect hybrid structures into the data set, an additional 75 synthetic Sybil graphs were added.

Since it is a topology-only characteristic Sybil clusters recognition/classification task, there are no real features, i.e. every node has a number of 1 assigned. Data have been split into training, validation and testing sets using a stratified 5-fold. The proportion is around 1027 samples for training, 115 for validation, and 285 for testing. The model has been trained using mini-batching.

4) Main Algorithm: In this work, only EOA's are being considered. More about it in the discussion in Chapter IV. 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, maximally connected components (MCC) are found. Then GIN classifies the MCCs that satisfy $|MCC| > x \wedge |MCC| < y$, where x, y are arbitrarily selected. Addresses belonging to Sybil MCC are saved, i.e. they are given a phase 0 algorithm score. It must be noted that we use only maximally connected 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.

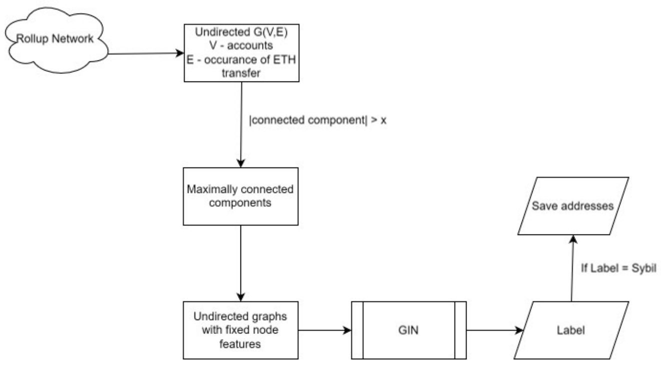
Rollup Network
Undirected G(V,E)
V - accounts
E - occurance of ETH transfer
|connected component| > x
Maximally connected components
Undirected graphs with fixed node features
GIN
Label
If Label = Sybil
Save addresses

Fig. 2. Binary classification algorithm

B. Phase One of The Framework

Phase 1 is the follow-up of [8], and we slightly modified their approach to our needs. Interactions involving smart contracts produce log entries which contain specific fields, called topics. In Ethereum based networks, topic0 refers to triggered event, such as transfer or swap, and each one is designated by a unique hash. On ZKsync, even a transfer of base token between EOAs, i.e. rollup's version of ETH, involves a system smart contract. For every account within MCC, a sequence with topic0 hashes is created, but in such a manner that if the same hash repeats, a dedicated iterator to that hash is concatenated with it. Then sets of Cartesian products of address event sequences are created with ordered pairs without reversal and without pairs in which elements are identical. Such products are then subject to comparison using the Jaccard coefficient (JC). Finally, a similarity matrix is built, which is then fed to the DBSCAN algorithm. Jaccard coefficient on sets A and B is calculated as follows:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{3}$$

Suppose that we have two addresses with very similar sequences, 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:

$$JC(A,B) = \frac{(n-k)(n-k-1)}{n(n-1)} \tag{4}$$

and assuming $\frac{k}{n} \to \infty$:

$$\lim_{n,k\to\infty} JC(A,B) = 1 \tag{5}$$

It 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 dedicated 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 $minPts = 2 \cdot dim$. 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 $minPts = \lfloor \sqrt{dim} \rfloor$. 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 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 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 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.

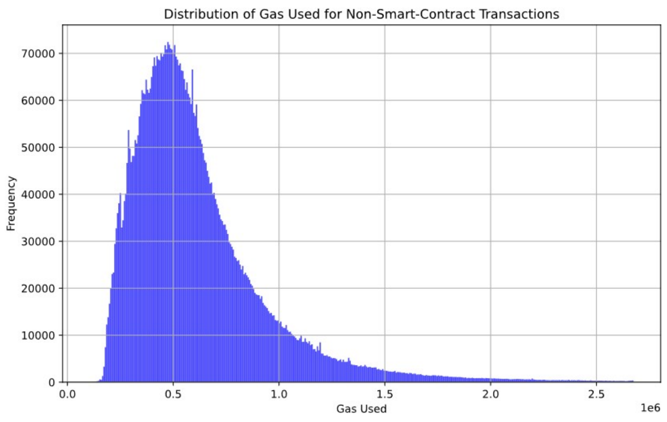

Fig. 3. Average gas usage distribution of EOAs

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:

$$Score = \frac{W_0 \cdot P_0 + W_1 \cdot P_1 + W_2 \cdot P_2 + W_3 \cdot P_3}{W_0 + W_1 + W_2 + W_3} \tag{6}$$

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

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.

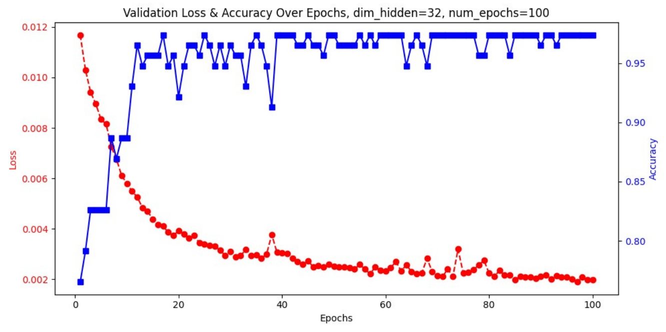

Fig. 4. Loss and accuracy over epochs in validation step

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.

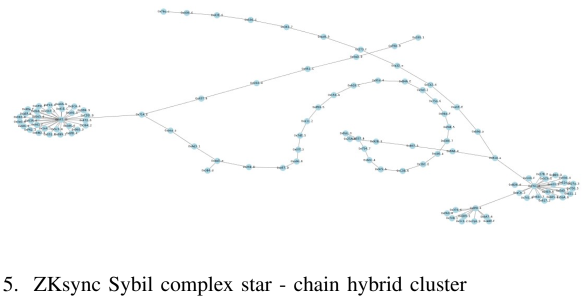

Fig. 5. ZKsync Sybil complex star - chain hybrid cluster

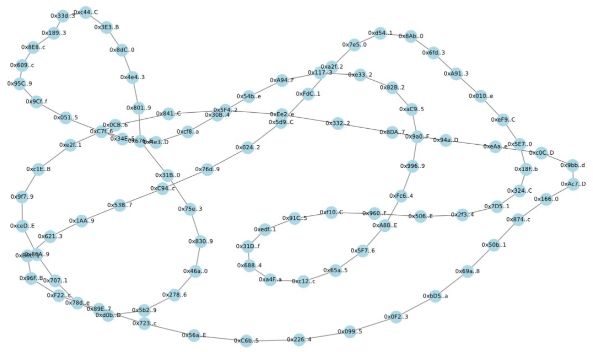

Fig. 6. ZKsync Sybil chain / cycle cluster

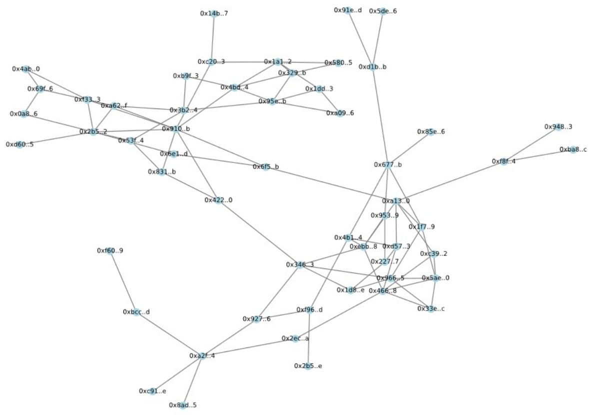

Fig. 7. ZKsync topologically normal cluster

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 randomness 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 represents 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:

$$\{a : (a \in MCC_0 \cap MCC_1) \wedge (\sum_{i=0}^{3} P_{ia} > 0)\}. \qquad (8)$$

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 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.

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%

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 contains 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

In this paper, many potential Sybil ZKsync accounts have been revealed. In practice, the proposed algorithm chain works like a sieve, where addresses with a higher score can be immediately excluded from airdrop, whereas those with a low score can be further investigated with different methods. It must be noted that filtering wallets too strictly, i.e. marking too many as Sybil (false positive), may discourage beneficial, honest users from the project. Our framework allows for the capture of the most prominent Sybil participants. The score threshold for classifying an address as malicious can be chosen arbitrarily, thus allowing us to be more flexible when it comes to false positive rate.

Quite a lot depends on the design of the airdrop. There exist airdrops with fixed token amount per user, but there can also be those, where the whole fixed pool of tokens is split between eligible addresses. That would mean that the more users are eligible, including Hunters, the fewer tokens each address receives, which makes it less cost-effective to perform Sybil attack. However, in such a design it would be more expected to find a poisoning attack in order to decrease competition.

It must also be reminded that there can exist online tutorials on how to become a partaker of airdrop, making many addresses perform similar actions on blockchain. Yet, our framework is also resistant to such phenomena to some extent, so honest users can gladly follow the same tutorial, unless their addresses are very close to each other in an address-transactions topology or if they belong to the same cluster. Moving on, there are also sites that make it easier to search for possible incoming airdrops [31]–[33], which can help Hunters choose the network to perform the Sybil attack. Developers who plan airdrop should be double cautious when their project is listed on such a site.

In terms of future work, a new approach to GIN training could be developed. In the event that more Sybil clusters are marked, one might be tempted to train GIN on directed graphs, which could further capture the nature of suspect clusters. Based on our observations during the development of this framework, we suspect that star topologies possessing main Sybil accounts, i.e. hubs, often have only outgoing directed edges from such central account, but not incoming edges. Another idea would be to train the machine learning model on account-level features and include it, perhaps, as another step in our framework. Here, a model would mark potential Sybil accounts based on it's behaviour, statistic attributes, like degree or measures of the significance of a vertex in a graph. Such an approach can potentially add a great contribution to the definition of characteristics of Sybil addresses.

A significant task would be to run our framework not only on datasets with EOAs but also with smart accounts. Detailed steps for creating such a dataset, including the filtering of some contracts and community detection, could be shown, thereby creating a great framework for blockchain data preparation.

Moreover, the proposed framework could be subject to comparison with different Sybil detection options, maybe on a relatively small labelled or synthetic data set. Such a comparison could show the characteristics of the results for every framework as well as the overall accuracy and similar metrics. It would allow projects to choose the most suitable approach.

REFERENCES

[1] The Block, "We made close to $1 million inside the murky world of airdrop farming," accessed: 2025-02-09. [Online]. Available: https://www.theblock.co/post/225215/we-made-close-to-1-million-inside-the-murky-world-of-airdrop-farming

[2] Binance Square, "Magic eden airdrop frenzy: One farmer cashes in over $2 million," accessed: 2025-02-09. [Online]. Available: https://www.binance.com/en/square/post/17443215929586

[3] X, accessed: 2025-02-09. [Online]. Available: 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.

[5] A. Yaish and B. Livshits, "Tierdrop: Harnessing airdrop farmers for user growth," *arXiv preprint arXiv:2407.01176*, 2024.

[6] TrustaLabs, "Airdrop-sybil-identification," accessed: 2025-03-13. [Online]. Available: https://github.com/TrustaLabs/Airdrop-Sybil-Identification

[7] Arbitrum Foundation, "Arbitrum sybil detection," accessed: 2025-03-13. [Online]. Available: https://github.com/ArbitrumFoundation/sybil-detection

[8] Z. Liu and H. Zhu, "Fighting sybils in airdrops," *arXiv preprint arXiv:2209.04603*, 2022.

[9] C. Sguanci, R. Spatafora, and A. M. Vergani, "Layer 2 blockchain scaling: A survey," *arXiv preprint arXiv:2107.10881*, 2021.

[10] ZK NATION, "Zk airdrop," accessed: 2025-04-13. [Online]. Available: https://docs.zknation.io/zk-token/zk-airdrop/

[11] K. Lommers, C. Makridis, and L. Verboven, "Designing airdrops," *Available at SSRN 4427295*, 2023.

[12] D. W. Allen, "Crypto airdrops: An evolutionary approach," *Journal of Evolutionary Economics*, vol. 34, no. 4, pp. 849–872, 2024.

[13] C. Zhou, H. Chen, H. Wu, J. Zhang, and W. Cai, "Artemis: Detecting airdrop hunters in nft markets with a graph learning system," in *Proceedings of the ACM Web Conference 2024*, 2024, pp. 1824–1834.

[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.

[15] J. Luo, H. Kang, S. Zheng, and X. Liu, "Toward resilient airdrop mechanisms: Empirical measurement of hunter profits and airdrop game theory modeling," *arXiv preprint arXiv:2503.14316*, 2025.

[16] S. Fan, T. Min, X. Wu, and W. Cai, "Altruistic and profit-oriented: Making sense of roles in web3 community from airdrop perspective," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 2023, pp. 1–16.

[17] T. Sato, M. Imamura, and K. Omote, "Threat analysis of poisoning attack against ethereum blockchain," in *IFIP International Conference on Information Security Theory and Practice*. Springer, 2019, pp. 139–154.

[18] T. Tsuchiya, J.-D. Dong, K. Soska, and N. Christin, "Blockchain address poisoning," *arXiv preprint arXiv:2501.16681*, 2025.

[19] Ethereum, "Erc-4337: Account abstraction using alt mempool," accessed: 2025-04-13. [Online]. Available: https://eips.ethereum.org/EIPS/eip-4337

[20] Zksync, accessed: 2025-04-13. [Online]. Available: https://docs.ZKsync.io/ZKsync-era/guides/ZKsync-101/paymaster

[21] Ethereum, "Eip-712: Typed structured data hashing and signing," accessed: 2025-04-13. [Online]. Available: https://eips.ethereum.org/EIPS/eip-4337

[22] Zksync, accessed: 2025-04-13. [Online]. Available: https://docs.ZKsync.io/ZKsync-protocol/account-abstraction/paymasters

[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.

[24] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.

[25] M. I. Silva, J. Messias, and B. Livshits, "A public dataset for the zksync rollup," *arXiv preprint arXiv:2407.18699*, 2024.

[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.

[27] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[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.

[29] A. Starczewski, P. Goetzen, and M. J. Er, "A new method for automatic determining of the dbscan parameters," *Journal of Artificial Intelligence and Soft Computing Research*, vol. 10, 2020.

[30] M. N. Gaonkar and K. Sawant, "Autoepsdbscan: Dbscan with eps automatic for large dataset," *International Journal on Advanced Computer Theory and Engineering*, vol. 2, no. 2, pp. 11–16, 2013.

[31] Alpha Drops, accessed: 2025-04-06. [Online]. Available: https://www.alphadrops.net/

[32] Bankless, accessed: 2025-04-06. [Online]. Available: https://www.bankless.com/airdrop-hunter

[33] Airdrops.io, accessed: 2025-04-06. [Online]. Available: https://airdrops.io/