Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2024.0429000

CENSor: Detecting Illicit Bitcoin Operation via GCN-based Hyperedge Classification

SUYEOL LEE[1], JAEHAN KIM[2], MINJAE SEO[3], SEUNG HO NA[2], SEUNGWON SHIN[2], and JINWOO KIM[4]

[1]FuriosaAI, Seoul, South Korea
[2]School of Electrical Engineering, KAIST, Daejeon, South Korea
[3]ETRI, Daejeon, South Korea
[4]School of Software, Kwangwoon University, Seoul, South Korea

Suyeol Lee and Jaehan Kim contributed equally.
Corresponding author: Jinwoo Kim (e-mail: jinwookim@kw.ac.kr).

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2021-0-00118, Development of decentralized consensus composition technology for large-scale nodes).

ABSTRACT Cryptocurrencies have increasingly been used as a medium for illicit financial activities by criminals. Annually, billions of dollars' worth of Bitcoin penetrate cryptocurrency exchanges. Despite the critical need for advanced Bitcoin financial forensics to investigate these criminal activities, no novel methods have been developed to detect illicit Bitcoin operations. Existing approaches to identifying illegal Bitcoin activity are fundamentally limited due to their inadequate consideration of graph data. To address these limitations, we present a novel approach, Hyperedge Classification, to detect illegal transactions with greater precision. This approach introduces a novel cluster-based Hyperedge-Node Switching technique, which enables effective hyperedge classification and visualization of hyperedge relationships. Additionally, we propose a framework named CENSor, which offers more powerful and robust detection capabilities compared to traditional techniques for both illegal entity detection and illegal transaction detection. Our cluster-based Hyperedge-Node Switching technique demonstrates a tenfold improvement in the F1-score compared to previous graph-based methods. Moreover, CENSor visualizes the Bitcoin cluster graph and the Hyperedge-Node switched graph, highlighting the importance of utilizing appropriate graph information in Bitcoin analysis.

INDEX TERMS Cryptocurrency, Illicit entity detection, Hypergraph

I. INTRODUCTION

Cryptocurrencies, such as Bitcoin [1] and Ethereum [2], distinguish themselves from fiat currencies by restructuring their foundational architecture and financial services. These cryptocurrencies are typically created through decentralized or distributed computing and are governed by majority consensus. Additionally, cryptocurrency transactions require only the involvement of two or related parties, eliminating the need for an intermediary service provider, such as a bank. Furthermore, cryptocurrencies generally offer pseudonymity, thereby ensuring the privacy of each participant.

The distributed architecture and pseudonymity inherent in cryptocurrencies also incentivize cybercriminals to adopt them as their primary currencies in cyberspace. These two features – distributed platforms and pseudonymity – render it challenging to identify the issuers of transactions and the holders of cryptocurrencies, thereby complicating efforts to analyze the behaviors of cryptocurrencies used for illicit purposes. Recent studies have demonstrated that Bitcoin is predominantly utilized in malicious operations on the Dark Web [3], and that cryptocurrency serves as a major currency in illegal market trading [4]–[6].

Indeed, the trend of adopting cryptocurrencies for cybercrimes is on the rise, prompting researchers and practitioners to develop detection systems to combat these illegal activities. For example, Vasek et al. [7] analyzed Bitcoin Ponzi fraud and surveyed scam trials in Bitcoin. Additionally, several proposals have been made to analyze Bitcoin abuses, including its use as a payment method for ransomware [8]–[11] and for dark market trading [3], [6], [12], [13]. However, despite the urgency of addressing the illicit use of cryptocurrencies, existing proposals have critical limitations. First, recent proposals do not adequately represent the overall illicit behaviors associated with cryptocurrencies. They tend to focus on a few specific features rather than considering the diverse aspects of cryptocurrency, resulting in a lack of comprehensive understanding of cryptocurrency behavior. Second, these proposals fail to consider the interconnected characteristics of cryptocurrencies. For instance, most cryptocurrencies can be represented as graph structures, which can reveal various interesting features (e.g., relationships among illicit Bitcoin addresses and transactions). Nonetheless, existing studies have not effectively utilized this graph structure in detecting illicit operations involving cryptocurrencies.

Other studies have employed graph representation learning to uncover illicit Bitcoin usage [14], [15]. However, they have commonly encountered two fundamental limitations when applying graph embedding techniques: *scalability* and *performance*. The Bitcoin address graph contains 700 million nodes and 4.8 billion edges, presenting an immense scale that is too large for the effective application of graph-based learning methods. Due to this scalability issue, prior efforts in classifying Bitcoin addresses have relied on outlier-based anomaly detection [16], [17], general unsupervised machine learning [18]–[22], and supervised machine learning [23]–[27] without incorporating graph information. Moreover, research related to illicit transaction detection, which involves the fine-grained analysis of illegal money flows, has recently garnered significant attention. Several studies regarding illicit Bitcoin flow detection have utilized the transaction graph [3], [14], [15], [21], [28]–[32]. However, the application of graph embedding methods alongside UTXO-based transaction graphs has suffered from low detection performance [14], [15], [32].

Goal and Approach. In this paper, we propose a graph-based framework, CENSor, to detect illicit Bitcoin activities by identifying illicit Bitcoin addresses and transactions[1]. Our design focuses on robust and high-performance illicit Bitcoin detection, addressing the fundamental challenges encountered in previous studies. Specifically, we formulate this detection task as a *hyperedge classification problem* to represent the diverse characteristics and features of the Bitcoin graph. Then, we introduce *cluster-based Hyperedge-Node Switching*, a novel technique to tackle the scalability and performance problems inherent in the hyperedge structure. This technique enables us to efficiently approximate the similarity between Bitcoin addresses and identify those associated with illicit activities. The experimental results demonstrate the effectiveness of our approach in detecting illicit Bitcoin addresses and transactions, significantly surpassing existing methods.

Summary and Contributions. By utilizing more meaningful (hyper)graph information, CENSor achieves robustness and high detection performance against adversarial attacks. Additionally, comparing the cluster-based Hyperedge-Node switched graph with the UTXO-based transaction graph enables us to understand the superior performance of our framework. The main contributions of this paper are summarized as follows:

- We introduce a novel illicit Bitcoin activity detection method, CENSor, which accurately captures the characteristics of Bitcoin transactions by transforming illicit transaction detection into hyperedge classification on the Bitcoin address graph (Section III and IV).
- We propose a hyperedge classification method specialized in visualizing hyperedge relation by using the Hyperedge-Node Switching technique (Section IV-D).
- We are the first to show the application of graph embedding on the billion-scale Bitcoin address graph for illicit entity detection. We propose a cluster-based illicit entity detection method that effectively addresses the challenges of scalability and performance in this context. (Section V-B and V-C).
- We visualize the cluster graph and Hyperedge-Node switched graph to provide insights on *"why (hyper)graph relations are important."* (Section VI-D and VI-C)
- CENSor achieves high performance and robustness in detecting illicit transactions (Section VI-C and VII).

II. BACKGROUND AND MOTIVATION

In this section, we describe key concepts of the most popular digital cryptocurrency, Bitcoin, and the state-of-the-art in tracing illegal activities of cryptocurrency. We then discuss the major drawbacks of prior projects, including motivating examples describing how adversaries easily incapacitate the current perimeters.

A. FINANCIAL FORENSICS IN A BITCOIN ENVIRONMENT

Bitcoin is the most popular cryptocurrency, relying on a distributed consensus protocol proposed by Satoshi Nakamoto [1]. Unlike the traditional banking system, the absence of a central authority results in pseudonymous Bitcoin-related financial activities. Additionally, anyone can freely generate cryptocurrency accounts and use them for any purpose without restrictions.

These advantages make Bitcoin attractive to adversaries for illegal financial activities, as it effectively conceals their identities. According to a report [34], $2.8 billion worth of Bitcoins flowed from criminal organizations to exchanges in 2019. The pseudonymous nature of cryptocurrency complicates the detection and tracing of illegal financial activities [35].

Financial forensics in the Bitcoin environment involves identifying illegal financial activities. Two major trends are: 1) non-graph-based solutions for detecting illicit entities, and 2) graph-based solutions for classifying illicit transactions.

[1]Since Bitcoin is known as the most dominant cryptocurrency for illicit operations [33], our primary focus is on detecting the illicit usage of Bitcoin.

1) Non-Graph-Based Solutions

Non-graph-based machine learning (ML) has been widely applied to identify illicit Bitcoin entities. With the intuition that illicit entities should have distinguishable behavioral anomalies, existing studies have focused on representative behavioral characteristics from identified illicit entities and applied traditional ML algorithms to reveal unknown illicit entities. Without labeled entities, the unsupervised ML algorithms, such as k-means clustering, Mahalanobis distance-based anomaly detection, and unsupervised SVM, are applied in prior systems [18]–[21]. Supervised ML models were widely adopted to uncover illicit Bitcoin entities, showing higher performance [23]–[27]. Harlev et al. [26] used the Gradient Boosting algorithm to predict the type of unidentified entities. Yin et al. [25] estimated the portion of cyber-criminal entities in uncategorized Bitcoin addresses with supervised machine learning models (e.g., Random Forest, Gradient Boosting classifiers, and MLP).

2) Graph-Based Solutions

The public decentralized ledger can be expressed as a graph structure consisting of financial transactions and associated addresses, which represents how much BTCs have flown to who/where. The Bitcoin financial transactions can be represented in multiple forms of graphs: the Bitcoin address graph, the Bitcoin hypergraph, and the UTXO-based transaction graph.

The *Bitcoin address graph* [36] is a data structure that expresses the relationships between addresses; two addresses are linked together when a transaction T exists between them. In the example shown in Figure 1a, $[A_1, A_2]$ represents the input address list and $[A_3, A_4]$ represents the output address list of T_1. In this situation, we can derive a Bitcoin address graph by linking all possible input-output pairs, as illustrated in Figure 1b.

The *Bitcoin hypergraph* is a data structure expressing the multiple nodes that are associated with a single transaction. As illustrated in Figure 1c, generated edges $((A_1, A_3), \cdots, (A_2, A_4))$ of the Bitcoin address graph from a single transaction (T_1) should be combined into one hyperedge. Therefore, the Bitcoin hypergraph considers the multiple edges generated from a single transaction as a single hyperedge unlike the Bitcoin address graph.

The *UTXO-based transaction graph (U-graph)* is a directed acyclic graph representation form of Bitcoin transactions. A Bitcoin transaction is generated from inputs and unspent transaction outputs (UTXOs). In other words, UTXOs are temporary Bitcoin chunks owned by a specific Bitcoin address that can be used as the input of a new transaction. As shown in Figure 1d, if the output UTXO of any transaction T_1 was used as input UTXO of transaction T_2, we can see that there was a flow in Bitcoin chunks through T_1 and T_2. Thus, we can trace Bitcoin flow by linking T_1 and T_2.

Previous studies relied on these graph-based solutions to trace black money flows [3] and detect suspicious transaction parties [14], [15], [37]. Lee et al. [3] presented a taint-based

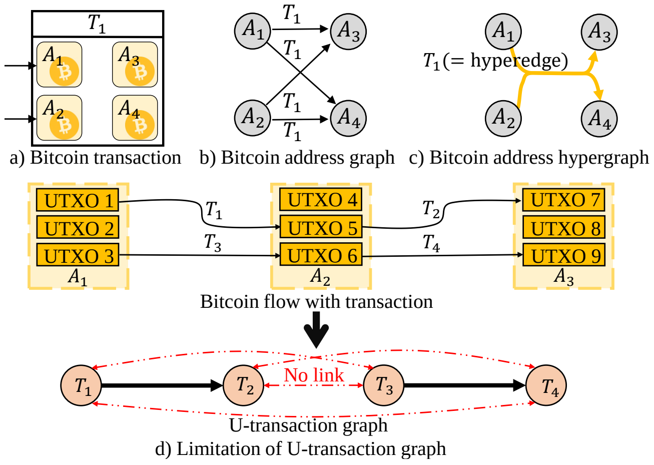


FIGURE 1: Bitcoin transaction graph

Bitcoin flow analysis to quantify the transferred Bitcoin volume and identify hidden information belonging to each transaction party. Hu et al. [15] adopted deepwalk and node2vec algorithms for graph embedding. Weber et al. [14] applied graph convolutional networks (GCNs) by using Bitcoin transactions as a feature for graph embedding.

B. LIMITATIONS OF EXISTING STUDIES

We observe that previous approaches have several challenges: i) *illicit entity detection* and ii) *illicit transaction detection*.

Challenge I: Illicit Entity Detection. Existing non-graph-based solutions [18]–[21], [23]–[27] mainly focus on individual behavioral features to classify illicit addresses. The reason for not utilizing graph-based techniques is that applying GCN or other graph embedding methods to the Bitcoin address graph is nearly impossible since the number of Bitcoin addresses and edges is enormous (e.g., 700M nodes and 4.8B edges). As a result, adversaries can exploit non-graph-based solutions by slightly adjusting individual address/entity features to make them similar to benign addresses/entities.

Figure 2a illustrates a scenario where an AML (Anti-Money Laundering) regulator employs non-graph-based ML to classify whether an address is illegal or not. Considering that non-graph-based ML relies only on individual features of an address, such as address balance and transaction fee, adversaries can evade the detection by slightly changing their individual features. As shown in Figure 2a, adversaries can create their addresses and transactions by imitating the feature of benign addresses. Non-graph-based ML models will recognize the illegal address imitating the licit address as benign; illegal transaction approaches also have the same issue.

Challenge II: Illicit Transaction Detection. Like illicit address/entity detection, illegal transaction detection must consider the features of related transactions within the graph structure; if a Bitcoin transaction is involved in illicit operations, it is likely to be involved in other illicit activities [9], [38]. This intuition is commonly used to uncover unknown

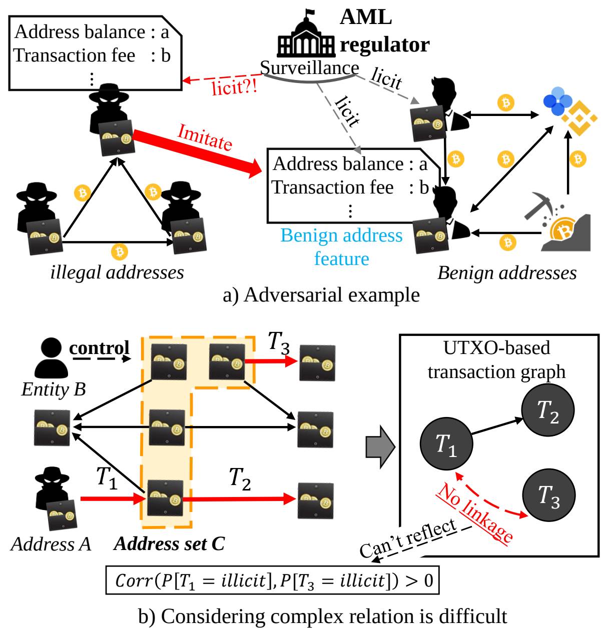


FIGURE 2: Motivating examples

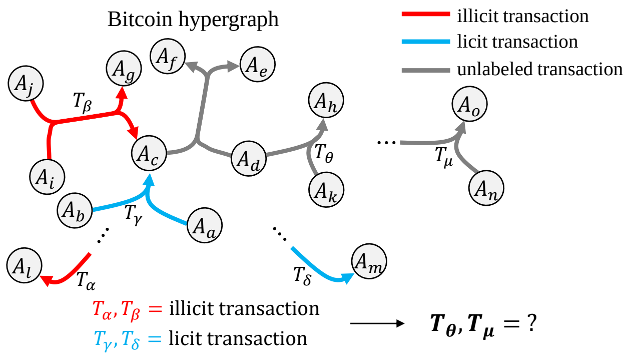


FIGURE 3: Problem statement for illicit transaction

illicit Bitcoin transactions. Consequently, previous studies utilized UTXO-based transaction graph to perform relational analysis with graph-based ML [14], [15], [37].

However, existing UTXO-based transaction graphs are unsuitable because they do not consider address-transaction relationships and, therefore, have incomplete information. In Figure 2b, address A transmits illegal transaction T_1 to address set C controlled by entity B. Transactions sent by entity B are likely to be illegal. Thus, the detection method for illicit transactions should predict a correlation between T_1 and T_3 as larger than 0, meaning that $Corr(P[T_1 = illicit], P[T_3 = illicit]) > 0$. However, the UTXO-based transaction graph does not have such links between T_1 and T_3, and therefore disregards the correlation [2], meaning that $Corr(P[T_1 = illicit], P[T_3 = illicit]) \approx 0$.

C. OUR APPROACH

To address the challenges, we present address clustering and our novel hyperedge classification method as the respective solutions.

Approach I: Bitcoin Address Clustering. The challenges for illicit entity detection is that applying graph embedding techniques to the billion-scale Bitcoin address graph is nearly impossible. For this, we summarized the Bitcoin address graph into the Bitcoin cluster graph to reduce the graph scale. Then, we apply GCN on the Bitcoin cluster graph to classify entities.

[2]Since GCN is inductive learning, it is possible that learned knowledge from T_1 could be used for classifying T_3 meaningfully even though T_1 and T_3 are in different weak connected component. However, this is applicable if there is no significant change in graph form and node features as time changes.

Approach II: Hyperedge Classification. We posit that utilizing the Bitcoin hypergraph (Section II-A2) facilitates more effective detection of illicit transactions compared to UTXO-based transaction graphs. However, the complexity of hyperedge relations presents classification challenges, making traditional edge classification methods inapplicable. To address this, we introduce a novel hyperedge classification method capable of handling these complexities.

III. PROBLEM FORMULATION

We begin by formulating the problem of illegal transaction detection as an edge classification task in the Bitcoin address graph. Given a Bitcoin hypergraph shown in Figure 3, our goal is to classify the unlabelled transactions T_θ and T_μ (i.e., illicit or licit), when existing transactions are known (i.e., T_α, T_β for illicit, and T_γ, T_δ for licit).

Here, we use a hyperedge classification method to classify unknown transactions. To formalize this problem, we define a Bitcoin address graph to be $G = (V, E)$, where V is a set of Bitcoin addresses, E is a set of transactions between addresses, and $E_l \subseteq E$ is the subset of edges labeled as either illicit or licit. Then, we define the binary edge classification problem as follows:

Definition 1 (Binary Edge Classification Problem). *Given a directed graph $G = (V, E)$ and a set of labeled edges $E_l \subseteq E$ where each edge $(u, v) \in E_l$ has a binary label $l_{uv} \in \{0, 1\}$, the objective of the binary edge classification problem is to determine the labels for the edges in $E_U = E - E_l$.*

Table 1 summarizes notations used in our formulation.

A. STRUCTURAL SIMILARITY MODEL

The issue of edge classification was first formulated and addressed by Aggarwal et al. [39]. They suggested a new structural similarity model in edge classification using a weighted Jaccard coefficient-based similarity metric. For this model, let $I^A(u) \subseteq V$ be the set of nodes incident to edges of u belonging to edge label A. Then, similarities between the two nodes are defined as follows:

$$J^0(u, v) = \frac{|I^0(u) \cap I^0(v)|}{|I^0(u) \cup I^0(v)|}, \quad J^1(u, v) = \frac{|I^1(u) \cap I^1(v)|}{|I^1(u) \cup I^1(v)|}$$

TABLE 1: Notations used in this paper

Notation	Meaning
$G = (V, E)$	Bitcoin address graph
V	Bitcoin addresses
E	Bitcoin transactions
$I^{label}(u)$	Set of nodes incident to *label*-label edges of u
$J^{label}(u, v)$	*label*-label similarity of two nodes u and v
$f^{label}(u, v)$	*label*-label fraction of nodes
$J(u, v)$	Jaccard similarity between two nodes u and v
$S(u)$	Set of top-k most similar nodes to u
E_l	Set of labeled edges
E_U	Set of unlabeled edges
$E_s(u, v)$	Labeled edges that linking nodes in $S(u)$ and $S(v)$
$s_{label}(u, v)$	Score of edge (u, v) to be labeled as *label*
$NE(u, v)$	Related edges of (u, v)
$W(u, v)$	Weight importance function
$l(u, v)$	Label of edge (u, v)
$l_p(u, v)$	Probability that label of edge (u, v) is 1
$f_{(u,v)}$	Feature of edge (u, v)
w	Learn-able weight vector in GCN
θ	Classification weight in classifier
G'	Edge-Node switched graph
V'	Part of Bitcoin transaction
E'	V' is linked with similarity set that incorporates the part of incoming/outgoing transaction
G_S	Hyperedge-Node switched graph
V_S	A transaction
E_S	V_S is linked with similarity set that incorporates the incoming/outgoing transaction

Then we define the fraction of nodes, $f^{label}(u, v)$, as:

$$f^{label}(u, v) = \frac{1}{2}\left(\frac{|I^{label}(u)|}{\sum_{i\in\{0,1\}} |I^{i}(u)|} + \frac{|I^{label}(v)|}{\sum_{i\in\{0,1\}} |I^{i}(v)|}\right)$$

Now weighted Jaccard similarities between u and v, $J(u, v)$, are summarized as:

$$J(u, v) = f^0(u, v) \cdot J^0(u, v) + f^1(u, v) \cdot J^1(u, v) \quad (1)$$

B. NAIVE METHOD FOR EDGE LABEL PREDICTION

Using the mentioned weighted similarities of nodes, the label of an unlabeled edge $(u, v) \in E_U$ can be predicted in the following steps.

- **Step 1**: Establish the top-k most similar nodes to u, denoted by $S(u) = \{u_1, u_2, \dots, u_k\}$ based on weighted Jaccard coefficient defined in Equation 1. $S(u)$ is the similarity set of u.
- **Step 2**: Establish the top-k most similar nodes to v, denoted by $S(v) = \{v_1, v_2, \dots, v_k\}$ based on weighted Jaccard coefficient defined in Equation 1. $S(v)$ is the similarity set of v.
- **Step 3**: Select the set of edges in $E_s(u, v) = E_l \cap [S(u) \times S(v)]$, and let $E_s^0(u, v) \subseteq E_s(u, v), E_s^1(u, v) \subseteq E_s(u, v)$ be the subsets of the respective edge label.
- **Step 4**: Then, the score for each label class s_0, s_1 can be defined as: [3]

[3]For simplicity, we assume that there exists exactly one transaction between addresses. However, our algorithm could be applied in any condition.

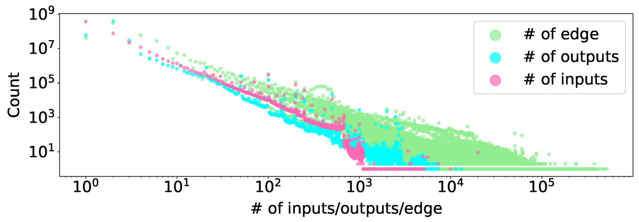


FIGURE 4: The distribution for the number of inputs, outputs, and edges generated from each transaction (in Feb. 22, 2020)

$$s_{label}(u, v) = \sum_{(u',v')\in E_s^{label}(u,v)} J(u, u') \cdot J(v, v') \quad (2)$$

Finally, if $s_0(u, v) > s_1(u, v)$, then the label for (u, v) is predicted as 0. Otherwise, the label is predicted as 1.

C. CHALLENGES IN BITCOIN EDGE LABEL PREDICTION

The naive method described above can be applied to the detection of illicit transactions on the Bitcoin graph, considering that $\forall(u, v) \in E$ are parts of transactions. However, when calculating the $s_0(u, v)$ and $s_1(u, v)$ on a Bitcoin address graph, we face five critical challenges.

C1. Scalability Issues for Finding the Similarity Set. The naive method requires a time complexity of $O(N^2)$ for calculating similarity among all nodes, with N being the number of nodes. For the Bitcoin graph, which includes nearly 700 million nodes, this calculation overhead is enormous.

C2. Neglecting Bitcoin Transaction Flow. The label score $s_{label}(u, v)$ defined in Equation 2 is calculated from the labeled edges in $S(u) \times S(v)$, which does not include information about the incoming or outgoing edges of $S(u)$ and $S(v)$. However, these edges represent Bitcoin transactions associated with money flow. Thus, if the incoming transactions of similarity set A are illicit, then the outgoing transactions of similarity set A are also likely to be illicit.

C3. Absence of Edge Weights. The label score s_{label} defined in Equation 2 sums similarities without considering importance. Some transactions, such as those with high output volume, might have a larger influence. Introducing a new weight vector corresponding to importance during supervised learning can address this issue.

C4. Insufficient Labeled Edges. Effective prediction of unknown Bitcoin addresses or transactions relies on some previously known information. However, collecting known benign or malicious Bitcoin address/transaction information is challenging. Predicting the label of edge (u, v) is also difficult when no labeled edges exist in $S(u) \times S(v)$ using similarity-based edge classification.

C5. Neglecting Hyperedges. Each Bitcoin transaction can generate a varying number of edges, constituting a *hyperedge*. Figure 4 depicts the distribution of inputs, outputs, and the number of edges for each transaction, clearly showing that the graph is a hypergraph.

IV. CLUSTER-BASED HYPEREDGE-NODE SWITCHING

To address the challenges, we propose a *time-efficient approximation algorithm* for calculating pairs of similarity nodes efficiently (C1) and a *new label score* to reflect Bitcoin transaction flow (C2) and account for the absence of edge weights (C3). Then, we devise a semi-supervised learning approach to handle insufficient labeled edges (C4). Finally, we introduce cluster-based Hyperedge-Node switching, which transforms the hyperedges of the original graph into nodes of a Hyperedge-Node switched graph (C5) and apply GCN.

A. APPROXIMATION FOR FINDING THE SIMILARITY SET

There are two general approaches to determine a pair of similarities: (i) direct calculation and (ii) locality-sensitive hashing (LSH). Here, we assume each node has edges with a substantial number of nodes within a two-hop distance. The time complexity of direct calculation is $O(N^2)$, while the time complexity of a recent LSH algorithm (e.g., c-Approximate r-Near Neighbor) is $O(N^{1+\frac{1}{c^2}}k)$, where k is the number of hash functions (typically, k is over 50) [40]. However, given that the Bitcoin address graph has 700 million nodes, we need a more efficient algorithm to find the similarity set in steps 1 and 2 of Section III-B.

Multi-Input Clustering. To address this problem, we employ the Bitcoin multi-input clustering [41], [42], which guarantees $O(M)$ time complexity, where M is the number of edges on the Bitcoin address graph. Additionally, it has been observed that the Bitcoin address graph has not densified since its first five years of existence [43], indicating that the density ($\alpha = M/N$) has remained constant. Therefore, the time complexity of multi-input clustering simplifies to $O(M) = O(\alpha N) \approx O(N)$.

Thus, multi-input clustering is an appropriate substitute for approximating a similarity set because it produces meaningful correlations. For example, if a_1, a_2, a_3 are input addresses of T_1 and a_2, a_3, a_4 are input addresses of T_2, then a_1, a_2, a_3, a_4 form the same cluster C_1. The input addresses of T_1 share the same edges as the output addresses of transactions. Therefore, for $u, v \in$ inputs of T_1, it guarantees the condition:

$$|I^{label}(u) \cap I^{label}(v)| \geq |outputs\ of\ T_1|$$

In addition, the definition of $J(u,v)$ is:

$$J(u,v) \propto |I^{label}(u) \cap I^{label}(v)|$$

and therefore $J(u,v)$ is guaranteed to be larger than a certain value only when u, v are from the same T_1, which becomes frequent when they are in the same cluster. This justifies using multi-input clustering as an approximation for finding the similarity set.

Table 2 summarizes time and memory complexities of each method, utilizing the multi-input clustering method being the most efficient. Therefore, we generate $S(u)$ and $S(v)$ utilizing multi-input clustering in the remainder sections.

TABLE 2: time complexity & memory complexity for deriving similarity set on each algorithm

	Direct	LSH [40]	Multi-input
Time complexity	$O(N^2)$	$O(N^{1+\frac{1}{c^2}}k)$	$O(N)$
Memory complexity	$O(N^2)$	$O(NlogN)$	$O(N)$

B. NEW LABEL SCORE FOR BITCOIN TRANSACTIONS

The label score, s_{label}, defined in Equation 2, only considers the labeled edges between $S(u)$ and $S(v)$. Consequently, the incoming and outgoing transactions of the similarity set are neglected and not taken into consideration in the calculation of s_{label}. Given that transactions represent flows of Bitcoin, the transactions from and to the similarity set are likely to be of a similar type, potentially offering valuable information in the label score s_{label}.

Weight Importance Function. Thus, the influence of incoming and outgoing transactions of a similarity set on s_{label} must be recognized. To address this, we introduce a weight importance function in calculating the label score. We then rewrite $s_{label}(u,v)$ and propose a new label score, $s^{new}_{label}(u,v)$, which incorporates the weight importance function $W : (u', v') \rightarrow \mathbb{R}$:

$$\begin{aligned} s^{new}_{label}(u,v) = & \sum_{(u',v') \in E^{label}_l, v' \in S(u)} W(u',v') \cdot J(u,v') \\ & + \sum_{(u',v') \in E^{label}_l, u' \in S(v)} W(u',v') \cdot J(v,u') \\ & + \sum_{(u',v') \in E^{label}_s(u,v)} W(u',v') \cdot J(u,u') \cdot J(v,v') \end{aligned} \tag{3}$$

In Equation 3, the first/second term is the label score related to incoming/outgoing transaction. In order to present how we achieve this new label score, we show how the original score is reshaped to the new score in Figure 5a. The score $s_0(u,v)$ indicates the extent to which the edge (u,v) relates to edges of label 0, and $s_1(u,v)$ provides a score for its relation to edges of label 1. $s^{new}_0(u,v)$ and $s^{new}_1(u,v)$ denote new label scores that consider relations of incoming and outgoing transaction.

C. APPLYING SEMI-SUPERVISED LEARNING

Sparsity Issue. To calculate $s^{new}_{label}(u,v)$, the sparsity issue must be addressed along with a method to optimize weight importance function $W : (u', v') \rightarrow \mathcal{R}$. For example, in Figure 5b, (a,b) is a labeled edge and (c,d) and (e,f) are unlabeled edges. Previous algorithms [39] could not calculate $s^{new}_{label}(e,f)$ because (e,f) has no linked labeled edge.

Semi-Supervised Learning. For this, we can predict the label score of $\forall(x,y) \in E$ by predicting 1) probability $l_p(x,y)$ that label value is 1 and 2) $W \cdot J$ for all edges including unlabeled edges. Edges can propagate their l_p to related edges [4]. Then, we can calculate $s^{pred}_{label}(u,v)$, the predicted value of $s^{new}_{label}(u,v)$,

[4]The related edges of (u,v) is $NE(u,v) = \{(u',v') \mid v' \in S(u)\} \cup \{(u',v') \mid u' \in S(v)\} \cup \{(u',v') \mid u' \in S(u)\ and\ v' \in S(v)\} \cup (u,v)$.

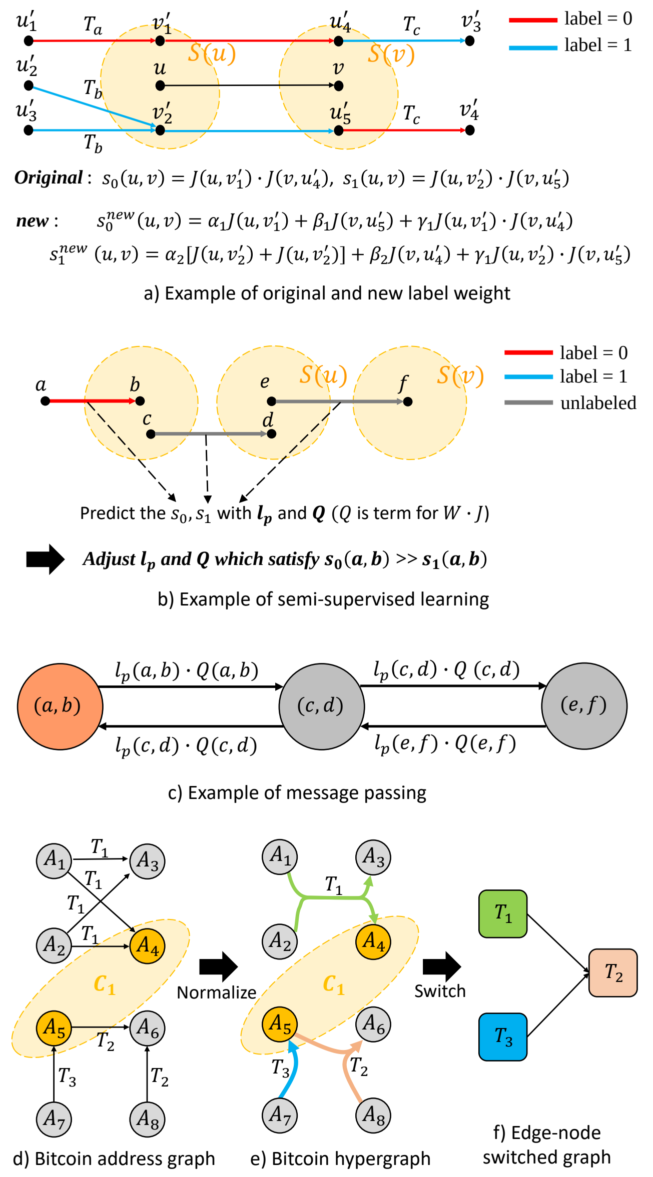

label = 0
label = 1
unlabeled
$S(u)$
$S(v)$
Original : $s_0(u,v) = J(u,v_1') \cdot J(v,u_4')$, $s_1(u,v) = J(u,v_2') \cdot J(v,u_5')$
new : $s_0^{new}(u,v) = \alpha_1 J(u,v_1') + \beta_1 J(v,u_5') + \gamma_1 J(u,v_1') \cdot J(v,u_4')$
$s_1^{new}(u,v) = \alpha_2 [J(u,v_2') + J(u,v_2')] + \beta_2 J(v,u_4') + \gamma_1 J(u,v_2') \cdot J(v,u_5')$
a) Example of original and new label weight
Predict the s_0, s_1 with l_p and Q (Q is term for $W \cdot J$)
Adjust l_p and Q which satisfy $s_0(a,b) >> s_1(a,b)$
b) Example of semi-supervised learning
$l_p(a,b) \cdot Q(a,b)$
$l_p(c,d) \cdot Q(c,d)$
$l_p(c,d) \cdot Q(c,d)$
$l_p(e,f) \cdot Q(e,f)$
c) Example of message passing
Normalize
Switch
C_1
d) Bitcoin address graph
e) Bitcoin hypergraph
f) Edge-node switched graph

FIGURE 5: Cluster-based Hyperedge-Node switching technique overview

by taking the multiplied sum of l_p of related edges with weight $Q = W \cdot J$ as follows:

$$s_0^{pred}(u,v) = \sum_{(u',v') \in NE(u,v)} (1 - l_p(u',v')) \cdot Q(u',v')$$

$$s_1^{pred}(u,v) = \sum_{(u',v') \in NE(u,v)} l_p(u',v') \cdot Q(u',v')$$

We optimize l_p and Q to minimize loss ($\mathcal{L}$) between target label values of labeled edges ($l(i,j)$) and predicted label values ($P_{(i,j)}$) as follows (That is, we want l_p and Q that could predict known labeled edge accurately) [5].

$$\mathcal{L} = \sum_{(i,j) \in E_l} l(i,j) \cdot \log P_{(i,j)} + (1 - l(i,j)) \cdot \log(1 - P_{(i,j)})$$

$$, P_{(i,j)} := \frac{e^{s_1^{pred}(i,j)}}{e^{s_0^{pred}(i,j)} + e^{s_1^{pred}(i,j)}}$$

As presented above, all unlabeled edges can be predicted by anticipating a real label of labeled edges accurately regardless of whether the labeled edges are sparse or not. However, in the real world, edges have their own features, so it is necessary to consider them and specific learning methods are required. We discuss these concerns in Section IV-D.

D. HYPEREDGE-NODE SWITCHING TECHNIQUE AND GCN

We now explain how s_{label}^{pred} can be obtained using GCN. First, we describe the relationship between s_{label}^{pred} and GCN, followed by an explanation of the graph switch when applying GCN. By incorporating normalization in a hypergraph approach, we then define our final cluster-based Hyperedge-Node switched graph.

Relation between s_{label}^{pred} and GCN. The label score s_{label} is predicted by passing and aggregating messages (Figure 5c), which is very similar to the fundamental idea of GCN. In practice, edges have their own features; therefore, l_p is derived from the feature of the edge ($f_{(u,v)}$) and the learnable weight vector w'.

$$l_p = f_{(u,v)}^T w'^T$$

Then, label score s_1^{pred} can be rewritten as:

$$s_1^{pred}(u,v) = \sum_{(u',v') \in NE(u,v)} f_{(u',v')}^T w'^T Q(u',v')$$

By replacing $w'^T Q$ with $w^T \theta_1$ and introducing an activation function σ, we write a new version of label score $s_1'^{pred}(u,v)$:

$$s_1'^{pred}(u,v) = \sigma\Big(\sum_{(u',v') \in NE(u,v)} f_{(u',v')}^T w^T \Big) \theta_1 \tag{4}$$

The above equation shows that $s_1'^{pred}(u,v)$ is the exact same form as the label score calculated from GCN [6]. Thus, we utilize GCN to calculate the label score. Equation 4 can be extended to k-layer GCN.

Edge-Node Switching and GCN. A GCN-friendly equation form was obtained for the label score in Equation 4. Therefore, we consider $\forall(x,y) \in E$ as a node because Equation 4 is the formula of GCN for a node classification task. GCN can then be used to predict $s_{label}(u,v)$ on a cluster-based Edge-Node switched graph, $G' = (V', E')$, where V' represents

[5] We could apply this process on toy example of Figure 5 (b), and $s(e,f)$ could be predicted as follows:

$$s_0^{pred}(e,f) = Q_{(e,f)}(c,d) + Q_{(e,f)}(e,f) - s_1^{pred}(e,f),$$
$$s_1^{pred}(e,f) = l_p(c,d) \cdot Q_{(e,f)}(c,d) + l_p(e,f) \cdot Q_{(e,f)}(e,f)$$

To optimize l_p and Q, we should derive l_p and Q such that satisfy $P_{(a,b)} \approx 0$.

[6] Except that $s_1'^{pred}(u,v)$ has no normalization term.

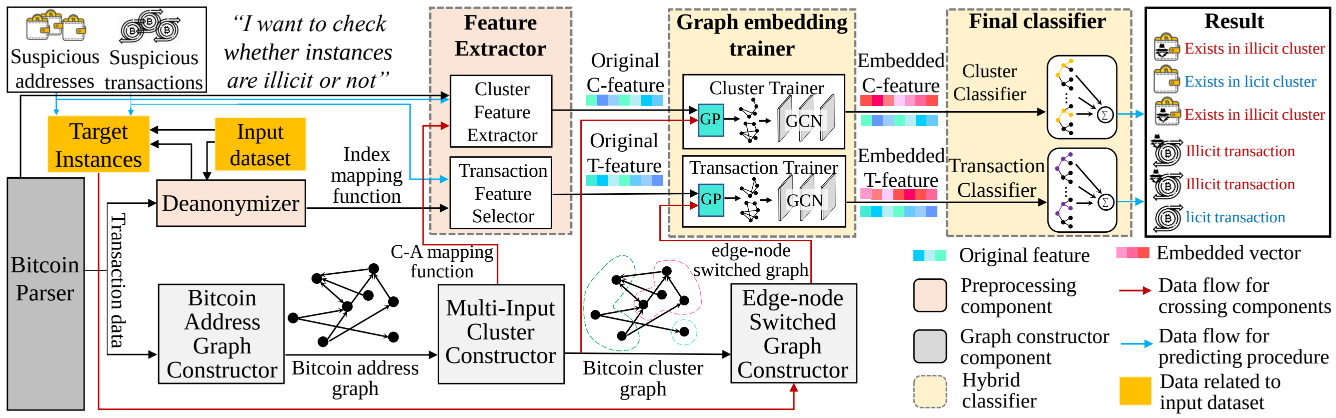


FIGURE 6: CENSor system overview

edges on the Bitcoin address graph, and E' is constructed between $\forall(x, y) \in E$ and $(x', y') \in NE(x, y)$.[7]

However, the Edge-Node switched graph G' has some issues because it treats the edges of the Bitcoin address graph[8] as independent objects from the transaction, even though the edges constructed from the same transaction must have the same characteristics (we lack information about each edge). Therefore, applying GCN to G' leads to problems of giving undue weight to edge-rich transactions and predicting different labels for edges constructed in the same transaction. To address this, we should view edges from the same transaction as a single object by introducing a cluster-based Hyperedge-Node Switched Graph.

Cluster-based Hyperedge-Node Switched Graph. In the Edge-Node switched graph, some transactions will contain many edges (of the Bitcoin address graph). If a hyperedge (transaction) T includes numerous edges $(x_1, y_1), \ldots, (x_n, y_n)$ and $\forall(x_i, y_i) \in NE(u, v)$ is established, the label score of (u, v) will become highly dependent on a single hyperedge (transaction) T. To reduce the dominating effect of these transactions on the label score, normalization must be considered in hyperedge classification. We set the hyperedge as the node of our Hyperedge-Node switched graph. This approach ensures that the number of edges generated from a particular hyperedge (transaction) will not affect the label score of our switched graph. Figures 5d, e, and f illustrate the cluster-based Hyperedge-Node switching technique.

To this end, we define the Hyperedge-Node switched graph (S-graph) $G_S = (V_S, E_S)$:

Definition 2 (S-Graph). *Let X be the set of hyperedges (transactions) and $C_{out}(y), C_{in}(y)$ be the set of multi-input clusters which include source and target nodes of hyperedge y. Then, an S-graph is defined as follows. $G_S = (V_S, E_S)$, where $V_S = \{y \mid y \in X\}$ and $E_S = \{(y, y') \mid C_{out}(y) \cap C_{in}(y') \neq \emptyset$ or $C_{out}(y') \cap C_{in}(y) \neq \emptyset\}$.*

[7]Importance between (a, b) and $(c, d) \in E_s(a, b)$ is considered by constructing a graph with a two-hop distance between these edges, considering that incoming and outgoing transactions of the similarity set have a closer relation.

[8]In this subsection, 'edge' always refers to the edge of the Bitcoin address graph, not the edge of the (Hyper)Edge-Node switched graph.

V. CENSor FRAMEWORK DESIGN

To assess the effectiveness of our method, we design and implement CENSor, a framework that processes graph data along with contextual data and trains a model performing illicit transaction detection and illicit cluster detection.

Framework Overview. Figure 6 shows the overview of CENSor, which comprises three main components: (i) the preprocessing components, (ii) the Graph constructor components, and (iii) the hybrid classifiers. The preprocessing component provides preprocessed non-graph data, such as the deanonymized elliptic dataset and cluster features for feature extraction. The graph constructor component builds the Bitcoin address graph, Bitcoin cluster graph, and the hyperedge-node switched graph needed for utilizing GCN. The data from these two components are then used to train the hybrid classification model.

Given that CENSor performs both illicit transaction and cluster detection, the hybrid classifier is divided into two modules: (i) the illicit transaction detection module and (ii) the illicit cluster detection module. The illicit transaction detection module is used for fine-grained detection of illicit Bitcoin flow, applying GCN on the Hyperedge-Node switched graph to perform hyperedge classification. The illicit cluster detection module is used for comprehensive illicit address detection. It addresses the scalability challenge posed by the large Bitcoin address graph by applying GCN on the Bitcoin cluster graph, which is a summarized version of the full Bitcoin address graph.

Below, we describe each component in detail.

A. PREPROCESSING COMPONENTS

1) Deanonymizer

CENSor uses the Elliptic dataset to evaluate its performance for illicit transaction detection. However, we cannot construct the Hyperedge-Node switched graph using only the anonymized Elliptic dataset. To address this, the deanonymizer matches the internal index of the Elliptic dataset to the real hash value of transactions using anonymized transaction features. Since some anonymized transaction feature values correlate with the degree of the transaction, deanonymizing the Elliptic dataset was feasible [44].

2) Feature Extractor

Cluster Feature Extractor. The cluster feature extractor extracts features of address clusters. Previous studies [18], [20], [21], [26], [34], [45]–[47] focus on extracting i) accumulated Bitcoin in the address, ii) flow of Bitcoin in the address, iii) timestamps of active behavior, and iv) degree of addresses as features. Our method is similar to these studies, but with a focus on cluster classification rather than address classification. CENSor extracts 77 features about clusters using 8 distribution statistics (i.e., sum, avg, median, max, min, std, kurtosis, and skewness).

Transaction Feature Selector. Instead of extracting transaction features directly, CENSor selects transaction features from the Elliptic dataset, as the evaluation will be based on this dataset.

B. GRAPH CONSTRUCTION COMPONENTS

Bitcoin Address Graph Constructor. This component parses the input and output addresses of transaction data collected from January 3, 2009, to February 22, 2020, using BlockSci [48], and generates the Bitcoin address graph, which consists of 700 million nodes and 4.8 billion edges.

Multi-input Cluster Constructor. This component clusters addresses in the Bitcoin address graph. The input addresses of each transaction are clustered, generating the Bitcoin cluster graph. We focus on the mainstream of the Bitcoin graph, considering only clusters with more than two addresses, resulting in a graph with 56 million nodes and 131 million edges. This reduction in size makes the graph manageable. Additionally, the multi-input Bitcoin cluster graph is used for both illicit transaction and cluster detection.

Edge-Node Switched Graph Constructor. It generates the Hyperedge-Node switched graph with target transactions that we want to classify. We generated the graph by leveraging our multi-input cluster-based edge node switching technique (Section IV-D).

C. HYBRID CLASSIFIERS

1) Graph Embedding Trainer

Cluster-GCN. GNN is a powerful tool for supervised graph representation learning. However, GNN algorithms commonly struggle with large-scale graphs because they cannot load the entire graph information into the GPU, which has a maximum RAM capacity of several dozen GB. Even using the mini-batch method for the GPU requires information about k-hop distance nodes for a k-layer GCN [49], and the GPU cannot handle this large memory space. To address this issue, we use a scalable GCN algorithm, Cluster-GCN [49], which

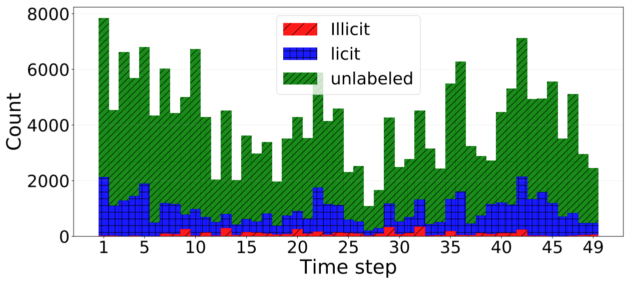


FIGURE 7: The Elliptic dataset over each time step

partitions the whole graph and uses these partitioned graphs as mini-batch training sets (We used Metis [50], which provides a balanced graph partition algorithm). By using Cluster-GCN, we encode the original feature into an embedded vector that represents how likely a node is to be illicit by optimizing the parameters.

Cluster Trainer. In the cluster trainer, embedded vector Z is generated from 3-layer GCN. Then, to optimize the weight matrix, W^0, W^1, W^2, we set the cross entropy loss function $\mathcal{L}$ for the supervised task with labeled illicit cluster dataset as follows, where $l(a)$ is the label value of address a and a_l are the set of labeled illicit cluster. σ is the softmax activation function and θ is the corresponding weight.

$$\mathcal{L} = \sum_{a \in a_l} l(a) \cdot log(\sigma(Z_a^T \theta)) + (1 - l(a)) \cdot (1 - log(\sigma(Z_a^T \theta)))$$

In training, we optimize θ, W^0, W^1, W^2 to minimize $\mathcal{L}$.

Transaction Trainer. In the transaction trainer, we learn weight vectors using methods similar to those used in the cluster trainer. The adjacency matrix is generated through the Hyperedge-Node switched graph, and learning is performed through a loss function calculated based on the feature and label values of the transactions.

2) Final Classifier

We append the final classifier to train the concatenated vector of the original feature and the embedded vector. We select Random Forest (RF) as the final classifier because it tends to outperform deep neural networks when the dataset is small or involves non-image data [51]–[53]. We construct a hybrid model by combining the GCN and RF classifiers, training the RF classifier on the concatenated vector of the original feature and the graph representation derived from the GCN.

VI. EVALUATION

In this section, we present our test methodology and the evaluation results.

A. DATASET PREPARATION

Bitcoin Data. We ran a Bitcoin Core version v0.19.99.0 to collect raw Bitcoin data between 01/2013 and 02/2020, and parsed raw data with BlockSci [48].

Scam Address Dataset. We collected scam addresses to construct a dataset of labeled addresses. The addresses were assembled from three sources: *i*) *Bitcoin blacklist*: The scam reports from two major Bitcoin scam blacklist services, BitcoinWhosWho [54] and BitcoinAbuse [55], *ii*) *Webpages*: Personal/technical blogs and scam analysis reports from cybersecurity companies, *iii*) *Address tags*: Meta-information about a Bitcoin address provided in Blockchain.com [9]. These addresses were labeled using natural language processing (NLP) techniques to analyze the context in which the addresses appeared.

After extracting and pruning, a total of 2,483 illicit addresses and 3,290 benign addresses were collected. We used random sampling to acquire the final dataset of 2,000 addresses for each label. The collected scam address data was used to label the Bitcoin address clusters. The clusters containing illicit addresses were considered to be illicit, while the clusters containing only licit addresses were named licit.

The Elliptic Dataset. To the best of our knowledge, the Elliptic dataset is the largest labeled Bitcoin transaction dataset, constructed by Elliptic, a cryptocurrency intelligence company [56]. The Elliptic dataset classifies transactions belonging to licit entities (e.g., wallet providers, exchanges, and miners) as *licit* transactions and transactions belonging to illicit entities (e.g., ransomware, scams, and malware) as *illicit* transactions.

The Elliptic dataset contains a total of 203,769 transactions: 4,545 illicit transactions, 42,019 licit transactions, and the remaining are unlabeled. Each transaction has 166 features consisting of local features and aggregated features. The local features describe individual transaction attributes, including time step, transaction fee, the numbers of inputs/outputs, and other specific characteristics. The aggregated features represent the collective attributes of one-hop transactions on the Hyperedge-Node switched graph. The dataset spans 49 time steps, each two weeks apart (Figure 7 shows the temporal distribution).

B. HYPERPARAMETER & TRAIN-TEST SET SPLIT

We set the hyperparameters and train/test set as follows:

Hyperparameters:

- *Illicit cluster detection.* We trained a 3-layer GCN model for 100 epochs with a learning rate of 0.0005 and node embedding size of 128.
- *Illicit transaction detection.* We trained the 3-layer GCN model for 300 epochs with a learning rate of 0.0005 and a node embedding size of 128.

Train/Test Set:

- *Illicit cluster detection.* We set the ratio of the train/test set as 0.2/0.8, and split train/test set randomly. The loss function weight was 0.5/0.5 ratio for each classes (illicit/licit).
- *Illicit transaction detection.* We set transactions from time steps 1-34 as the training set and those from time steps 35-49 as the test set. This setup allows us to assess how the models predict future data effectively. The loss function weight was 0.7/0.3 ratio for each classes (illicit/licit).

[9]https://www.blockchain.com

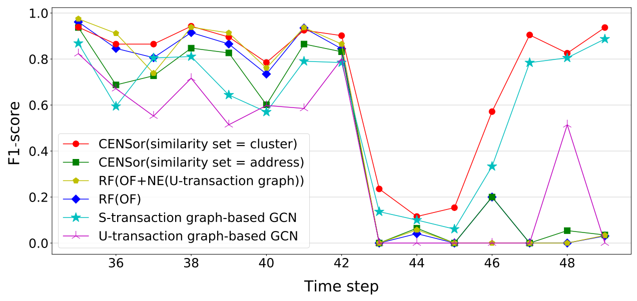


FIGURE 8: Illicit transaction F1-score over time step

TABLE 3: Illicit transaction classification result

Method	Precision	Recall	F1	Unweighted F1
CENSor (cluster)	0.901	**0.832**	**0.867**	**0.724**
CENSor (address)	0.923	0.623	0.744	0.445
GCN + S-graph	0.758	0.699	0.727	0.598
RF (OF & U-graph)*	**0.933**	0.708	0.805	0.482
RF (OF)	0.898	0.726	0.803	0.479

* This model is the same method used by Weber [14].

C. ILLICIT TRANSACTION DETECTION

Experiment. We evaluated the illicit transaction detection performance of our Hyperedge-Node switching graph (S-graph)-based GNN compared to Random Forest (RF) and UTXO-based graph (U-graph)-based GCN using the deanonymized Elliptic dataset. We did not consider other baseline models like MLP or logistic regression since [14] showed that these methods perform worse than RF.

To evaluate performance, we predicted all transactions in the Elliptic dataset. While existing edge classification methods can only predict the labels of transactions that share similarity sets with labeled transactions, CENSor can predict the labels of all transactions even if only 0.01% (46,000) of all Bitcoin transactions are labeled. An important consideration in assessing detection method is the stability. Thus, we compare the methods noting that the major dark markets were shut down in time step 43.

Detection Performance. In Figure 8, the illicit transaction F1-score of the method applying GCN on the S-graph and U-graph is represented by the cyan and magenta lines over time steps. As shown in Figure 8, the S-graph-based GCN outperforms the U-graph-based GCN, particularly after time step 43, where the performance of the S-graph-based GCN is significantly better.

Effect of Address Clustering. The method of selecting the similarity set can affect performance. To compare with CENSor, which approximates the similarity set by clusters, we constructed another version of CENSor that establishes a similarity set by addresses. As illustrated in Figure 8 and Table 3,

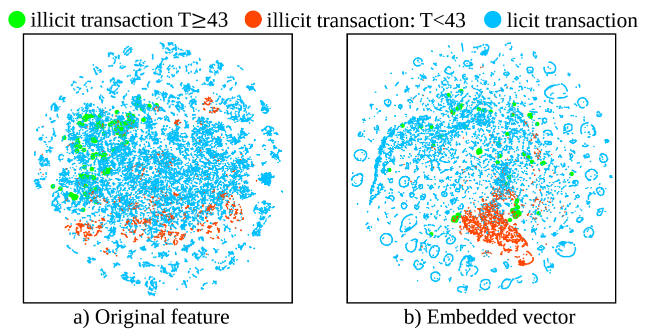


FIGURE 9: Visualization of transaction feature with t-SNE

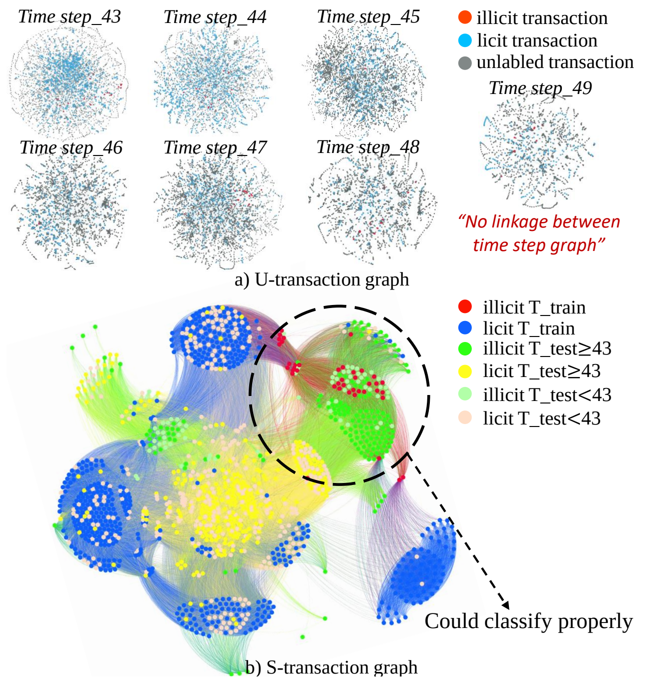


FIGURE 10: Visualization of U-graph and S-graph. a) U-transaction graph after time step 43. b) One-hop graph of illicit transactions (after time step 43) is visualized.

setting each address to a similarity set shows a noticeable decrease in performance. This difference highlights that the selection strategy of the similarity set is a crucial factor.

CENSor vs Traditional Methods. As shown in Figure 8, CENSor outperforms other methods after time step 43. Specifically, CENSor had a recall value of 0.832 (See Table 3), which was significantly higher than the U-graph-based approach of 0.708, meaning that CENSor could cover diverse illicit activities. In addition, we define *unweighted F1* as the unweighted average of F1-scores for each time step since the weighted F1-score underestimates the performance after time step 43; the number of transactions labeled illicit after time step 43 is small (see Figure 7). The *unweighted F1* of CENSor was 0.724, superior to the U-graph-based method with a score below 0.5.

Synergy Effect of Graph Information. To understand the significance and impact of utilizing graph information, we examine the synergy effect, defined by the performance improvement of graph-based models over non-graph-based models. The synergy effect of the U-graph-based method is 0.002 (=0.805 - 0.803) (see Table 3), whereas the synergy effect of CENSor is **0.064** (=0.867 - 0.803) (see Table 3), which is **30** times greater. This substantial difference demonstrates how effectively CENSor utilizes graph information. A detailed explanation of this improvement is provided in Section VI-C.

Visualization of Learned Graph Representation. CENSor uses learned graph representations from GCN to train an illicit transaction classification model. Figure 9 provides a visualization of the graph representation using t-SNE, a technique that effectively visualizes high-dimensional data in a two-dimensional space [57]. We observed that the representations of illicit transactions (test set after time step 43) learned by CENSor are more tightly clustered compared to those derived from original transaction features. This suggests that S-graph-based GCN could enhance methods that rely solely on original features.

Visualization of S-Graph. Since S-graph and U-graph could be matched one-to-one, CENSor is optimized to explain why hyperedge classification is superior compared to the U-graph-based method. Figure 10a illustrates U-graphs after time step 43 and shows that the graphs at each step are completely separate from each other. Since GCN is inductive learning, it might be trained well on an unseen graph. However, if the time step graph severely changes after some time step due to important events, the inductive learning of GCN will no longer show good performance. For this reason, the U-graph-based model has failed to detect illicit transactions after time step 43. However, Figure 10b shows that illicit transactions in our S-graph after time step 43 had relations with illicit transactions of the train set. This means that our hyperedge classification method could consider more difficult and complex relations, achieving high performance.

D. ILLICIT CLUSTER DETECTION

Experiment. We evaluated the illicit cluster detection performance of CENSor and existing baseline models on the scam address dataset. We selected Random Forest (RF), Gradient Boosting (GB), and Multi-layer Perceptron (MLP), which had shown good performance in previous studies [25], [26], as the baseline models. All models were trained with 68 features related to incoming and outgoing transactions. We then used embedded vectors produced during the epoch with the lowest training loss from the GCN models to train the final classifier. The mean and variance of F1-scores were measured over 10 trials for each model.

Detection Performance. As shown in Figure 11a, GCN outperforms RF and GB classifier, and the hybrid models (i.e., GB+GCN and CENSor) shows the best F1-score in all models. Thus, we could verify our hybrid model was the best method in illicit cluster detection.

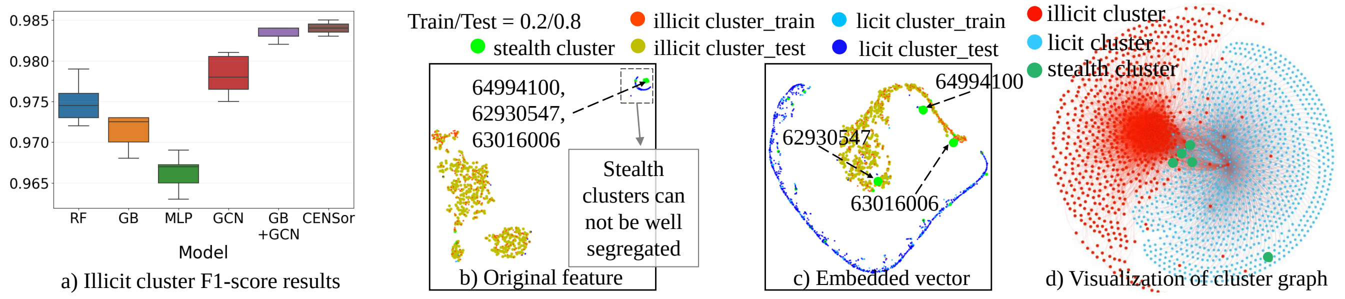


FIGURE 11: Illicit cluster F1-score (a), visualization of original feature/embedded vectors (b/c), and cluster graph (d)

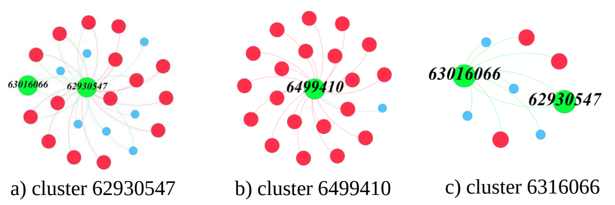


FIGURE 12: Subgraph for stealth cluster

Visualization of Learned Graph Representation. CENSor uses learned graph representation from GCN to train the final classifier. To show an effectiveness of the graph representation learned from GCN, we visualize the original node feature and the embedded node vector with t-SNE. As shown in Figure 11b and c, red/blue/green points represent illicit/licit/stealth clusters. It can be seen that the original features of licit clusters and stealth clusters are not well segregated compared to the results of the embedded vectors. For this reason, the GCN and hybrid models outperform other baseline methods.

Visualization of Clusters. To demonstrate that CENSor effectively captures graph information, we visualized the Bitcoin cluster graph using the visualization tool Gephi [58]. Figure 11d visualizes the cluster graph only with labeled clusters, as the full Bitcoin cluster graph is too large. As illustrated in Figure 11d, illicit clusters typically transmit Bitcoin among themselves, and licit clusters also conduct transactions with each other. This suggests that GCN, which aggregates the features of multi-hop neighborhood nodes, can perform better in detecting illicit clusters.

Case Study for Stealth Clusters. We identified *stealth clusters* that are undetectable by baseline models but detectable by CENSor. We analyzed an example of stealth clusters in detail for a deeper understanding. As shown in Figure 12, clusters 62930547, 6316066, and 64994100 have several transactions with illicit clusters (cluster numbers are our internal index). However, the baseline models classified clusters 62930547, 6316066, and 64994100 as licit. In contrast, GCN, which considers graph structure, could classify these clusters correctly. The capability of GCN is demonstrated in Figures 11b and 11c. When we reduced the dimensionality of vectors using t-SNE, the original features of clusters 62930547, 6316066, and 64994100 exhibited values similar to those of licit clusters, while the embedded vectors learned from GCN showed similarity values aligned with illicit clusters. Thus, GCN could represent each node considering the information of their multi-hop neighborhood nodes, and CENSor demonstrates superior performance.

VII. ADVERSARIAL ATTACKS AND ROBUSTNESS

In this section, we evaluate the robustness of CENSor against adversarial attacks compared to other models in detecting illicit transactions. Here, an adversarial attack refers to an attempt to hide the attacker's illicit transactions from being detected. Our assumptions for the adversarial attack method are outlined in Section VII-A. Based on these assumptions, we demonstrate that it is extremely challenging for an attacker to alter the graph structure. Therefore, we concluded that an attacker would likely conduct an adversarial attack by simply changing the features of their illicit transactions. As described in Section VII-B, we modified the features of the attacker's illicit transactions using the Fast Gradient Sign Method (FGSM) to simulate adversarial attacks.

A. ASSUMPTIONS FOR ADVERSARIAL ATTACKS

There are several strategies to conduct adversarial attacks, and the adversary needs to decide on the method. To this end, we make three reasonable assumptions.

Assumption 1. *Modifying the graph structure of the Bitcoin address graph is challenging.* Since the Bitcoin address graph is generated through the POW consensus algorithm, modifying the Bitcoin address graph has several limitations:

- An adversary cannot remove a previously created transaction history once the block is confirmed.
- An adversary cannot disguise a future transaction as having been created before the present time.
- An adversary can alter the Bitcoin address graph structure by sending transactions to normal addresses, but only a small amount of Bitcoins can be sent since transmitted Bitcoins cannot be retrieved.
- An adversary cannot generate incoming transactions from addresses they do not control.

Thus, we assume that performing an adversarial attack by changing the Bitcoin address graph structure is nearly impossible.

TABLE 4: Illicit transaction F1-score of models when an adversary carries out an adversarial attack using FGSM. ϵ indicates how much adversaries can control their features for FGSM. Results are presented as *average* $\pm$ *std* over 10 trials (Embedded vectors were produced from the epoch with the lowest train loss).

ϵ	CENSor	Random Forest with U-graph + GCN*	Random Forest (RF)
0	**0.867±0.004**	0.805±0.004	0.803±0.006
0.001	**0.708±0.054**	0.517±0.105	0.642±0.147
0.002	**0.705±0.083**	0.544±0.074	0.439±0.261
0.005	**0.672±0.053**	0.475±0.079	0.251±0.246
0.01	**0.568±0.090**	0.218±0.079	0.004±0.005
0.02	**0.396±0.083**	0.095±0.069	0.000±0.000
0.05	**0.315±0.112**	0.020±0.026	0.000±0.000
0.1	**0.329±0.074**	0.000±0.000	0.000±0.000
0.2	**0.371±0.158**	0.000±0.000	0.000±0.000
0.5	**0.675±0.048**	0.000±0.000	0.000±0.000
1	**0.680±0.071**	0.000±0.000	0.000±0.000

* This model is the same method used by Weber [14].

Assumption 2. *An adversary can alter the features of their illicit transactions within a small range.* An adversary can adjust features such as outputs or fees of transactions they generate, within the limits of their intended purpose.

Assumption 3. *An adversary is fully aware of the data in the training set for the transaction monitoring system.* Since Elliptic data is publicly available and CENSor is evaluated using this data in the paper, we can assume that the adversary knows the data used to train CENSor.

B. SCENARIO AND METHOD FOR ADVERSARIAL ATTACKS

Based on the assumptions described in Section VII-A, we set up a black-box adversarial attack scenario:

- **Scenario Definition**: The adversary attempts to deceive the transaction monitoring system by fine-tuning the target model on the features of their illicit transactions in the training set.
- **Train Set**: Elliptic transaction data from time steps 1-34.
- **Adversary**: The adversary aims for their illicit transactions (i.e., those occurring in time steps 35-49) to be classified as licit transactions in the actual transaction monitoring system.
- **Target Model**: Since the adversary does not know which models are used in the transaction monitoring system, they conduct a black-box adversarial attack to manipulate the transaction features, targeting the most common classification model, the neural network.

Method for Adversarial Attacks. As described in Section VII-A, an adversary can only attack by adjusting the features of their transactions. We chose FGSM (Fast Gradient Sign Method) to perform adversarial attacks. FGSM is a fast and simple method for generating adversarial examples for neural networks [59]. FGSM first calculates the gradient of the cost function, $\nabla_x J(\theta, x, y)$. Subsequently, the sign value of the gradient, multiplied by the possible maximum value of noise (ϵ), is added to the original data feature to generate perturbed data x' as follows:

$$x' = x + \epsilon \cdot sign(\nabla_x J(\theta, x, y))$$

This results in a larger cost function calculated from the perturbed data features (x') than with the original features (x). We performed the FGSM attack about 2-layer neural network classifier with 50 neurons in each layer, and set the ϵ 0 to 1.

C. RESULTS FOR ROBUSTNESS OF CENSor

We compared the robustness of CENSor against adversarial attacks with that of the RF model and the RF model utilizing embedded vectors obtained by applying GCN to the U-graph. Table 4 shows the F1-score of each model as ϵ varies. The typical RF model converged to an F1-score of 0 when ϵ was greater than 0.02. The RF model, which utilizes embedded vectors obtained by applying GCN to U-graphs, began to converge to an F1-score of 0 when ϵ was greater than 0.05. However, CENSor maintained a high F1-score even for high values of ϵ, demonstrating its robustness against adversarial attacks.

This robustness occurs because CENSor aggregates the features of other transactions that share the same similarity sets in the graph structure. In other words, CENSor incorporates and reflects more features from related transactions compared to the previous method of applying GCN to the U-graph, where some loss of graph information is inevitable. Even if the individual features of the transactions are altered, the overall effect is minimized, resulting in increased robustness.

VIII. DISCUSSION AND LIMITATION

Performance across Time Steps. The evaluation of CENSor shows high performance compared to previous works. Although it achieved the highest F1-score during time steps 43-45, the method's performance was significantly lower during these time steps compared to others. Figure 8 shows that CENSor has F1-scores around 0.2 during this period, while achieving scores between 0.8 and 1 at other time steps. This low F1-score is due to the fact that a few misjudgments can result in poor scores, given that there are only a dozen illicit transactions in time steps 43-45 (see Figure 7).

Robustness for Adversarial Attacks. The transactions in time steps 43-45 represented a new type of transaction that had no connection with the transactions used for training. In addition to high performance, CENSor is more robust against adversarial attacks, even when attackers change their features with a large noise level (ϵ). If attackers severely alter their features with a large ϵ, the perturbed illicit transactions also change the aggregated transaction features accordingly. Therefore, if A stands for perturbed transactions and B represents transactions in the training set, then transactions A and B, which share the same similarity set, will exhibit new types of features compared to other transactions. As a result, the embedded vectors of A and B would be similar.

Limitations in Bitcoin Address Clustering. However, CENSor shows a limitation in the accuracy of Bitcoin address clustering. If payments from multiple Bitcoin senders are combined into one transaction using a method like CoinJoin, different senders can be clustered together. This limitation may also affect the illicit cluster detection module of CENSor. When illicit addresses and other user addresses form a cluster through CoinJoin, we can consider them as a *suspicious cluster*. A suspicious cluster is a broad category that may include illicit addresses, but it is challenging to determine which specific addresses within the suspicious cluster are exactly illicit.

Impact on Illicit Transaction Detection. This limitation, however, does not affect illicit transaction detection. Unlike illicit cluster detection, multi-input clustering is used to approximate the similarity set for illicit transaction detection. Since the similarity set and ownership of addresses are irrelevant in this context, using multi-input clustering for illicit transaction detection presents no disadvantages.

IX. RELATED WORK

Bitcoin Crime Analysis. Many criminals have begun exploiting Bitcoin to hide their illicit activities. Several studies have analyzed Bitcoin abuse in various forms, such as ransomware [8], [9], scams [7], [60], and dark market trading [3], [6], [12], [13].

Bitcoin Address Classification. The early method of Bitcoin address classification involved clustering Bitcoin addresses to known (previously labeled) addresses and classifying addresses in the same cluster into the same category [61]. The application of machine learning has brought forth classification methods with higher performance, utilizing supervised learning techniques such as random forest and boosting methods [25], [26]. Some studies used graphical features to enhance their models with relational information between nodes, but these efforts did not fully utilize the graph information and were limited to features such as the proportion of certain transaction motif patterns [62], [63].

Bitcoin Transaction Classification. Due to the complexity of the edge classification task and the intuition that a Bitcoin transaction resembles an edge in the Bitcoin address graph, very few studies exist on Bitcoin transaction classification. Previous studies on Bitcoin transaction classification use the U-transaction graph to regard the transactions as nodes [14], [15], [37]. Hu et al. [15] used node2vec as the graph embedding method, and Weber et al. [14] utilized GCN in the U-transaction graph. Lorenz et al. [37] proposed a method that can be applied even if the labeled data is small through active learning. Despite these efforts, the U-graph fails to properly represent and utilize all the relational information that graph data can offer. These drawbacks have been overcome by our novel method of transforming the transaction graph into a Hyperedge-Node switched graph.

Illicit Financial Activity Detection. Research on detecting accounts engaged in illicit activities is prominent in services dealing with cryptocurrency. Wu et al. [64] proposed an embedding method using an algorithm that incorporated the volume and timestamp of transactions in a node2vec-based algorithm to detect accounts used for Ethereum phishing scams. Chen et al. [65] suggested detecting Ethereum phishing accounts through cascade feature extraction using a LightGBM-based dual-sampling method. Tam et al. [66] applied GCN with vectors concatenating node and edge features to detect illicit accounts in Ethereum scam data and Tencent mobile payment transaction data. Liu et al. [67] used an attention vector-based heterogeneous graph neural network to detect malicious accounts in the Alipayment system. Multi-view-based deep learning methods are also gaining attention. Zhong et al. [68] introduced a multi-view meta-path attention-based financial defaulter detection method in the Alibaba payment system. Tao et al. [69] used multi-view graph attention networks to detect real money trading in NetEase MMORPG. However, existing works detect illicit users, accounts, or addresses based on node classification tasks. To the best of our knowledge, this is the first work to detect illicit financial activities by leveraging an edge classification approach, effectively representing such activities.

Edge Classification. The issue of edge classification has always been important, but it has only recently been properly formulated [39], [70], [70]–[72]. Aggarwal et al. [39] suggested a method using weighted Jaccard similarity for edge classification. However, their method could not handle the sparsity issue. To solve this problem, Gupta et al. [70] proposed a subgraph projection-based edge classification method. Their method is applicable only in signed graphs and cannot be expanded to the problem of hyperedge classification. Song et al. [73] classified relation types (e.g., friend and family) in the WeChat network through a community classification-based edge prediction method. However, they classify static relationships within social networks, which is inappropriate for classifying transactions, a monetary flow.

X. CONCLUSION

We design CENSor, a framework that detects illegal Bitcoin transactions and illegal Bitcoin clusters, overcoming the limitations of previous Bitcoin forensics methods. Due to the difficulties posed by the application of existing edge classification methods to illegal transaction detection, we propose a novel hyperedge classification method that utilizes GCN on the S-graph, a graph constructed by our new cluster-based Hyperedge-Node switching technique. By properly utilizing the information that the graph offers, the performance of models following CENSor is significantly higher than that of previous studies in detecting illegal transactions.

CENSor also demonstrates that using GCN in detecting illegal addresses can improve detection performance. We tested our own model using FGSM adversarial attacks and demonstrated the exceptional robustness of CENSor compared to legacy methods. CENSor shows that the legitimate utilization of graph information can lead to higher performance and robustness in financial forensics models.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Manubot, Tech. Rep., 2019.

[2] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[3] S. Lee, C. Yoon, H. Kang, Y. Kim, Y. Kim, D. Han, S. Son, and S. Shin, "Cybercriminal minds: An investigative study of cryptocurrency abuses in the dark web." in *NDSS*, 2019.

[4] "Wannacry, petya, notpetya," https://www.theguardian.com/technology/2017/dec/30/wannacry-petya-notpetya-ransomware.

[5] "Silk road," https://en.wikipedia.org/wiki/Silk_Road_(marketplace).

[6] S. Foley, J. R. Karlsen, and T. J. Putniņš, "Sex, drugs, and bitcoin: How much illegal activity is financed through cryptocurrencies?" *The Review of Financial Studies*, vol. 32, no. 5, pp. 1798–1853, 2019.

[7] M. Vasek and T. Moore, "There's no free lunch, even using bitcoin: Tracking the popularity and profits of virtual currency scams," in *International conference on financial cryptography and data security*. Springer, 2015, pp. 44–61.

[8] K. Liao, Z. Zhao, A. Doupé, and G.-J. Ahn, "Behind closed doors: measurement and analysis of cryptolocker ransoms in bitcoin," in *2016 APWG Symposium on Electronic Crime Research (eCrime)*. IEEE, 2016, pp. 1–13.

[9] M. Paquet-Clouston, B. Haslhofer, and B. Dupont, "Ransomware payments in the bitcoin ecosystem," *Journal of Cybersecurity*, vol. 5, no. 1, p. tyz003, 2019.

[10] J. Choi, J. Kim, M. Song, H. Kim, N. Park, M. Seo, Y. Jin, and S. Shin, "A large-scale bitcoin abuse measurement and clustering analysis utilizing public reports," *IEICE TRANSACTIONS on Information and Systems*, vol. 105, no. 7, pp. 1296–1307, 2022.

[11] C. Patsakis, E. Politou, E. Alepis, and J. Hernandez-Castro, "Cashing out crypto: state of practice in ransom payments," *International Journal of Information Security*, vol. 23, no. 2, pp. 699–712, 2024.

[12] N. Christin, "Traveling the silk road: A measurement analysis of a large anonymous online marketplace," in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 213–224.

[13] H. Al Jawaheri, M. Al Sabah, Y. Boshmaf, and A. Erbad, "Deanonymizing tor hidden service users through bitcoin transactions analysis," *Computers & Security*, vol. 89, p. 101684, 2020.

[14] M. Weber, G. Domeniconi, J. Chen, D. K. I. Weidele, C. Bellei, T. Robinson, and C. E. Leiserson, "Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics," *arXiv preprint arXiv:1908.02591*, 2019.

[15] Y. Hu, S. Seneviratne, K. Thilakarathna, K. Fukuda, and A. Seneviratne, "Characterizing and detecting money laundering activities on the bitcoin network," *arXiv preprint arXiv:1912.12060*, 2019.

[16] D. D. F. Maesa, A. Marino, and L. Ricci, "Detecting artificial behaviours in the bitcoin users graph," *Online Social Networks and Media*, vol. 3, pp. 63–74, 2017.

[17] M. A. Prado-Romero, C. Doerr, and A. Gago-Alonso, "Discovering bitcoin mixing using anomaly detection," in *Iberoamerican Congress on Pattern Recognition*. Springer, 2017, pp. 534–541.

[18] P. M. Monamo, V. Marivate, and B. Twala, "A multifaceted approach to bitcoin fraud detection: Global and local outliers," in *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2016, pp. 188–194.

[19] P. Monamo, V. Marivate, and B. Twala, "Unsupervised learning for robust bitcoin fraud detection," in *2016 Information Security for South Africa (ISSA)*. IEEE, 2016, pp. 129–134.

[20] T. Pham and S. Lee, "Anomaly detection in bitcoin network using unsupervised learning methods," *arXiv preprint arXiv:1611.03941*, 2016.

[21] ——, "Anomaly detection in the bitcoin system a network perspective," *arXiv preprint arXiv:1611.03942*, 2016.

[22] D. Chaudhari, R. Agarwal, and S. K. Shukla, "Towards malicious address identification in bitcoin," in *2021 IEEE international conference on blockchain (Blockchain)*. IEEE, 2021, pp. 425–432.

[23] M. Bartoletti, B. Pes, and S. Serusi, "Data mining for detecting bitcoin ponzi schemes," in *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*. IEEE, 2018, pp. 75–84.

[24] Y.-J. Lin, P.-W. Wu, C.-H. Hsu, I.-P. Tu, and S.-w. Liao, "An evaluation of bitcoin address classification based on transaction history summarization," in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 2019, pp. 302–310.

[25] H. S. Yin and R. Vatrapu, "A first estimation of the proportion of cybercriminal entities in the bitcoin ecosystem using supervised machine learning," in *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 2017, pp. 3690–3699.

[26] M. A. Harlev, H. Sun Yin, K. C. Langenheldt, R. Mukkamala, and R. Vatrapu, "Breaking bad: De-anonymising entity types on the bitcoin blockchain using supervised machine learning," in *Proceedings of the 51st Hawaii International Conference on System Sciences*, 2018.

[27] J. Liang, L. Li, S. Luan, L. Gan, and D. Zeng, "Bitcoin exchange addresses identification and its application in online drug trading regulation," 2019.

[28] D. Ron and A. Shamir, "Quantitative analysis of the full bitcoin transaction graph," in *International Conference on Financial Cryptography and Data Security*. Springer, 2013, pp. 6–24.

[29] F. Reid and M. Harrigan, "An analysis of anonymity in the bitcoin system," in *Security and privacy in social networks*. Springer, 2013, pp. 197–223.

[30] M. Moser, "Anonymity of bitcoin transactions," 2013.

[31] M. Möser, R. Böhme, and D. Breuker, "An inquiry into money laundering tools in the bitcoin ecosystem," in *2013 APWG eCrime Researchers Summit*. Ieee, 2013, pp. 1–14.

[32] K. Martin, M. Rahouti, M. Ayyash, and I. Alsmadi, "Anomaly detection in blockchain using network representation and machine learning," *Security and Privacy*, vol. 5, no. 2, p. e192, 2022.

[33] C. C. Intelligence, "Cryptocurrency anti money laundring report," 2018.

[34] "Chain analysis," https://www.chainalysis.com/.

[35] *Bitcoin virtual currency: Unique features present distinct challenges for deterring illicit activity.*, FBI, 2012, https://bit.ly/2nuMpTl.

[36] E. Filtz, A. Polleres, R. Karl, and B. Haslhofer, "Evolution of the bitcoin address graph," in *Data science–Analytics and applications*. Springer, 2017, pp. 77–82.

[37] J. Lorenz, M. I. Silva, D. Aparício, J. T. Ascensão, and P. Bizarro, "Machine learning methods to detect money laundering in the bitcoin blockchain in the presence of label scarcity," *arXiv preprint arXiv:2005.14635*, 2020.

[38] Z. Samsudeen, D. Perera, and M. Fernando, "Behavioral analysis of bitcoin users on illegal transactions."

[39] C. Aggarwal, G. He, and P. Zhao, "Edge classification in networks," in *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. IEEE, 2016, pp. 1038–1049.

[40] P. Indyk, "Near optimal hashing algorithms for approximate near (est) neighbor problem," in *MMDS 2006. Workshop on Algorithms for Modern Massive Data Sets, Stanford, USA*, 2006.

[41] J. D. Nick, "Data-driven de-anonymization in bitcoin," Master's thesis, ETH-Zürich, 2015.

[42] M. Harrigan and C. Fretter, "The unreasonable effectiveness of address clustering," in *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*. IEEE, 2016, pp. 368–373.

[43] J. Liang, L. Li, and D. Zeng, "Evolutionary dynamics of cryptocurrency transaction networks: An empirical study," *PloS one*, vol. 13, no. 8, 2018.

[44] "Deanonymizing elliptic dataset transactions," https://geeks-world.github.io/articles/479178/index.html.

[45] A. Gaihre, Y. Luo, and H. Liu, "Do bitcoin users really care about anonymity? an analysis of the bitcoin transaction graph," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 1198–1207.

[46] J. Hirshman, Y. Huang, and S. Macke, "Unsupervised approaches to detecting anomalous behavior in the bitcoin transaction network," *3rd ed. Technical report, Stanford University*, 2013.

[47] A. Gaihre, S. Pandey, and H. Liu, "Deanonymizing cryptocurrency with graph learning: The promises and challenges," in *2019 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2019, pp. 1–3.

[48] H. Kalodner, S. Goldfeder, A. Chator, M. Möser, and A. Narayanan, "Blocksci: Design and applications of a blockchain analysis platform," *arXiv preprint arXiv:1709.02489*, 2017.

[49] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh, "Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 257–266.

[50] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM Journal on scientific Computing*, vol. 20, no. 1, pp. 359–392, 1998.

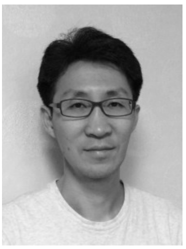
[51] M. Liu, M. Wang, J. Wang, and D. Li, "Comparison of random forest, support vector machine and back propagation neural network for electronic tongue data classification: Application to the recognition of orange beverage and chinese vinegar," *Sensors and Actuators B: Chemical*, vol. 177, pp. 970–980, 2013.
[52] L. Benali, G. Notton, A. Fouilloy, C. Voyant, and R. Dizene, "Solar radiation forecasting using artificial neural network and random forest methods: Application to normal beam, horizontal diffuse and global components," *Renewable energy*, vol. 132, pp. 871–884, 2019.
[53] C.-H. Hsieh, R.-H. Lu, N.-H. Lee, W.-T. Chiu, M.-H. Hsu, and Y.-C. J. Li, "Novel solutions for an old disease: diagnosis of acute appendicitis with random forest, support vector machines, and artificial neural networks," *Surgery*, vol. 149, no. 1, pp. 87–93, 2011.
[54] "Bitcoinwhoswho," https://bitcoinwhoswho.com/.
[55] "Bitcoinabuse," https://www.bitcoinabuse.com/.
[56] "Elliptic," www.elliptic.co.
[57] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.
[58] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: an open source software for exploring and manipulating networks," in *Proceedings of the international AAAI conference on web and social media*, vol. 3, no. 1, 2009, pp. 361–362.
[59] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6572
[60] M. Vasek and T. Moore, "Analyzing the bitcoin ponzi scheme ecosystem," in *International Conference on Financial Cryptography and Data Security*. Springer, 2018, pp. 101–112.
[61] M. Spagnuolo, F. Maggi, and S. Zanero, "Bitiodine: Extracting intelligence from the bitcoin network," in *International Conference on Financial Cryptography and Data Security*. Springer, 2014, pp. 457–468.
[62] S. Ranshous, C. A. Joslyn, S. Kreyling, K. Nowak, N. F. Samatova, C. L. West, and S. Winters, "Exchange pattern mining in the bitcoin transaction directed hypergraph," in *International Conference on Financial Cryptography and Data Security*. Springer, 2017, pp. 248–263.
[63] J. Wu, J. Liu, W. Chen, H. Huang, Z. Zheng, and Y. Zhang, "Detecting mixing services via mining bitcoin transaction network with hybrid motifs," *arXiv preprint arXiv:2001.05233*, 2020.
[64] J. Wu, Q. Yuan, D. Lin, W. You, W. Chen, C. Chen, and Z. Zheng, "Who are the phishers? phishing scam detection on ethereum via network embedding," *arXiv preprint arXiv:1911.09259*, 2019.
[65] W. Chen, X. Guo, Z. Chen, Z. Zheng, and Y. Lu, "Phishing scam detection on ethereum: Towards financial security for blockchain ecosystem," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020 [scheduled for July 2020, Yokohama, Japan, postponed due to the Corona pandemic]*, C. Bessiere, Ed. ijcai.org, 2020, pp. 4506–4512. [Online]. Available: https://doi.org/10.24963/ijcai.2020/621
[66] D. S. H. Tam, W. C. Lau, B. Hu, Q. F. Ying, D. M. Chiu, and H. Liu, "Identifying illicit accounts in large scale e-payment networks–a graph representation learning approach," *arXiv preprint arXiv:1906.05546*, 2019.
[67] Z. Liu, C. Chen, X. Yang, J. Zhou, X. Li, and L. Song, "Heterogeneous graph neural networks for malicious account detection," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 2077–2085.
[68] Q. Zhong, Y. Liu, X. Ao, B. Hu, J. Feng, J. Tang, and Q. He, "Financial defaulter detection on online credit payment via multi-view attributed heterogeneous information network," in *Proceedings of The Web Conference 2020*, 2020, pp. 785–795.
[69] J. Tao, J. Lin, S. Zhang, S. Zhao, R. Wu, C. Fan, and P. Cui, "Mvan: Multi-view attention networks for real money trading detection in online games," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 2536–2546.
[70] M. Gupta and R. Mishra, "Network projection-based edge classification framework for signed networks," *Decision Support Systems*, p. 113321, 2020.
[71] Z. Yang, W. Pei, M. Chen, and C. Yue, "Wtagraph: Web tracking and advertising detection using graph neural networks," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 1540–1557.
[72] J. Kim, M. Song, M. Seo, Y. Jin, and S. Shin, "Passrefinder: Credential stuffing risk prediction by representing password reuse between websites on a graph," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 2024, pp. 20–20.
[73] C. Song, Q. Lin, G. Ling, Z. Zhang, H. Chen, J. Liao, and C. Chen, "Locec: Local community-based edge classification in large online social networks," in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 2020, pp. 1689–1700.

SUYEOL LEE is a researcher at FuriosaAI. He received his M.S. degree and B.S. degree both in School of Electrical Engineering at KAIST. His research interests focus on the analysis of illicit bitcoin activities using graph-based deep learning.

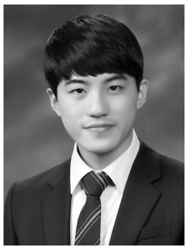
JAEHAN KIM is a PhD candidate in the School of Electrical Engineering at KAIST. He received his B.S. degree and M.S. degree in School of Electrical Engineering at KAIST. His current research interests mainly focus on Cyber security, Machine learning Security, Large language model and Data mining.

MINJAE SEO is a researcher at ETRI, Daejeon, South Korea. He received his M.S. degree from the Graduate School of Information Security at KAIST and his B.S. degree in Computer Engineering from Mississippi State University. His current research interests include Software-defined networking security, network fingerprinting, and deep learning-based network system.

SEUNG HO NA is a Ph.D. candidate in the Network and System Security Lab, advised by Seungwon Shin, in the School of Electrical Engineering at KAIST. He received his M.S. degree and B.S. degree in Electrical Engineering from KAIST. His research interests span the areas of data-driven security, artificial intelligence (AI) security, and cyber threat intelligence (CTI). Currently, he focuses on preserving and improving privacy of machine learning systems.



SEUNGWON SHIN is an Associate Professor in the School of Electrical Engineering at KAIST and an Executive Vice President at Samsung Electronics. He received his Ph.D. in Computer Engineering from the Electrical and Computer Engineering Department at Texas A&M University and his M.S. degree and B.S. degree from KAIST, both in Electrical and Computer Engineering. His research interests span the areas of Software-defined networking security, DarkWeb analysis, and Cyber threat intelligence.



JINWOO KIM is an Assistant Professor in the School of Software at Kwangwoon University, Seoul, South Korea. He received his Ph.D. from the School of Electrical Engineering at KAIST, his M.S. degree from the Graduate School of Information Security at KAIST, and his B.S. degree from Chungnam National University in Computer Science and Engineering. His research topic focuses on investigating security issues with software-defined networks and cloud systems.