TRacer: Scalable Graph-based Transaction Tracing for Account-based Blockchain Trading Systems

Zhiying Wu*
Jieli Liu*
wuzhy95@mail2.sysu.edu.cn
liujli7@mail2.sysu.edu.cn
Sun Yat-sen University
Guangzhou, Guangdong, China

Jiajing Wu†
wujiajing@mail.sysu.edu.cn
Sun Yat-sen University
Guangzhou, Guangdong, China

Zibin Zheng
zhzibin@mail.sysu.edu.cn
Sun Yat-sen University
Guangzhou, Guangdong, China

ABSTRACT

Security incidents such as scams and hacks, have become a major threat to the health of the blockchain ecosystem, causing billions of dollars in losses each year for blockchain users. To reveal the real-world entities behind the pseudonymous blockchain account and recover the stolen funds from the massive transaction data, much effort has been devoted to tracing the flow of illicit funds in blockchains recently. However, most current tracing approaches based on heuristics and taint analysis have limitations in terms of universality, effectiveness, and efficiency. This paper models the blockchain transaction records as a blockchain transaction graph and tackles blockchain transaction tracing as a graph searching task. We propose **TRacer**, a scalable transaction tracing tool for account-based blockchains. To infer the relevance between accounts during graph searching, we develop a novel personalized PageRank method in TRacer based on the directed, weighted, temporal, and multi-relationship blockchain transaction graphs. To the best of our knowledge, TRacer is the first intelligent transaction tracing tool in account-based blockchains that can handle complex transaction actions in decentralized finance (DeFi). Experimental results and theoretical analysis prove that TRacer can complete the transaction tracing task effectively at a low cost. All codes of TRacer are available at GitHub [1].

CCS CONCEPTS

• **Applied computing** → **Digital cash**; • **Mathematics of computing** → **Graph algorithms**.

KEYWORDS

Blockchain, transaction tracing, personalized PageRank, local community discovery

*Both authors contributed equally to this research.
†Corresponding author.
[1]https://github.com/wuzhy1ng/BlockchainSpider

KDD '22, August 14–18, 2022, Washington DC, USA

ACM ISBN 978-1-4503-XXXX-X/22/06…$15.00
https://doi.org/xx.xxxx/xxxxxxx.xxxxxxx

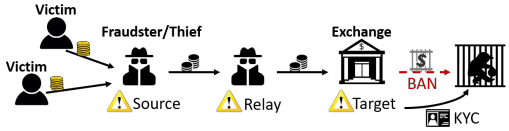
ACM Reference Format:
Zhiying Wu, Jieli Liu, Jiajing Wu, and Zibin Zheng. 2022. TRacer: Scalable Graph-based Transaction Tracing for Account-based Blockchain Trading Systems. In *Proceedings of ACM Conference (KDD '22).* ACM, New York, NY, USA, 10 pages. https://doi.org/xx.xxxx/xxxxxxx.xxxxxxx

1 INTRODUCTION

The rapid development of blockchain technology has aroused great attention of businesses and researchers recently. By incorporating peer-to-peer networks, cryptography, and consensus protocols, blockchain [28] achieves a decentralized environment for trading and brings new vitality to traditional industries. Particularly, the typical account-based blockchain platform Ethereum has opened the era of blockchain 2.0 through the introduction of smart contracts, giving blockchain various possibilities of application. However, the pseudonymous nature of blockchain has also attracted a variety of illegal transaction activities like financial scams and hacks. According to a recent report of Chainalysis [8], a famous blockchain security company, the losses caused by illegal transaction activities in cryptocurrency-related businesses have exceeded $14 billion during 2021. Along with the boom of DeFi, most of these illegal trading activities and malicious attacks are conducted in the account-based blockchain trading systems like Ethereum and Binance Smart Chain (BSC) [6] since the boom of DeFi.

With the publicly accessible blockchain transaction data, various technologies have been developed to combat financial crimes in blockchain trading systems [21], and these technologies can be divided into two categories, namely **proactive (pre-trade) risk warning** and **remedial (post-trade) money tracking**. Proactive risk warning refers to evaluating the risk of new transactions according to the historical behaviors of the related accounts and the existing label information. Data-based fraud detection [22], attack detection [24], and other types of illegal transaction detection technologies [19] can be categorized in this scope. However, as shown in Figure 1(a), although proactive early warning can raise warnings for risky transactions before the occurrence of the trade, it cannot prevent the criminals who has already gotten the money from laundering and cashing out of the ill-gotten gains from exchanges since the pseudonymous and distributed nature of blockchain systems. Therefore, a wealth of efforts have been devoted to the remedial money tracking of the ill-gotten gains [9, 23, 26], aiming to deanonymize the related money flows and help victims recover the losses.

(a) Proactive risk warning in blockchain



(b) Remedial money tracking in blockchain

Figure 1: (a) Proactive risk warning in blockchain raises warnings for risky transactions. (b) Remedial money tracking in blockchain traces the money flow among the source and the targets possessing the laundered funds, also providing evidence to catch the criminals off-chain.

Figure 1(b) shows a toy example of the remedial money tracing. Generally speaking, transaction tracing starts from a source and traces the flow of money to the targets. Here, the source represents a tracing object such as the blockchain account of a fraudster decamping with a large number of ill-gotten gains, and the targets indicate the accounts used to gather the "clean" funds awaiting cashing out. Though the identity information of accounts is unknown in blockchain systems, once we locate target deposit addresses of exchanges that enforce Know-Your-Customer (KYC) processes, the related criminals can be identified and caught off-chain according to the KYC information provided by the exchanges [20].

Current approaches for blockchain transaction tracing [14, 16, 18, 27] are mainly based on rule-based heuristics and taint analysis [13]. However, as an emerging research topic, existing transaction tracing methods have limitations in terms of universality, effectiveness and efficiency. Particularly, most existing heuristic methods are designed for specific scenarios based on experts experience, and cannot be automatically and intelligently applied to various blockchain transaction scenarios. In addition, the time cost and end conditions of existing methods, especially those requiring manual verification and intervention, is not definite or well defined, making their time efficiency and effectiveness difficult to guarantee. Besides, the popularity of DeFi in account-based blockchains like Ethereum and BSC brings many new kinds of semantics to blockchain transaction actions, leading to high barriers to the transaction tracing task.

In this paper, we propose the first general tool called **TRacer**, which is a transaction Tracing in account-based blockchain trading systems incorporating Personalized PageRank-based technologies. Referring to the framework displayed in Figure 2, TRacer first models the blockchain transaction data including the complex DeFi operation actions into a directed, weighted, temporal, and multi-relationship graph, and then conducts local community detection on a subgraph around the risky seed obtained by graph expansion. The output small-scale community is the traced money flow graph and can be further audited by experts. Moreover, we propose a novel ranking algorithm based on personalized PageRank [15] to reveal the relevance between the source and other accounts in blockchain systems. We introduce approximate personalized PageRank (APPR) [1, 2, 25] to obtain the approximate solution of personalized PageRank, which can improve the scalability of the algorithm. Both graph expansion and local community detection in TRacer are based on the proposed ranking algorithm. Experimental results demonstrate the performance of TRacer on transaction tracing in account-based blockchains. The main contributions of this work can be summarized as follows:

- To the best of our knowledge, we are the first to study intelligent transaction tracing in account-based blockchain trading systems like Ethereum, which is an urgent problem with the booming security incidents in these systems.
- We design and implement a general blockchain transaction tracing tool named TRacer, which is able to incorporate the complex semantics of transaction actions in DeFi. Particularly, a novel personalized PageRank method is employed to estimate the relevance score of accounts in TRacer.
- We thoroughly evaluate the performance of TRacer via both theoretical analysis and experimental evaluation, and the results demonstrate the effectiveness and the scalability of TRacer. We also contribute a benchmark dataset verified by several security companies for evaluating the transaction tracing methods.

2 RELATED WORK

To combat crimes in blockchain, transaction tracing is a critical task for blockchain security companies and the community. Current methods for blockchain transaction tracing are mainly designed for Bitcoin-like blockchain systems and heavily rely on expert experience. The most widely used transaction tracing method is Breath First Search (BFS) [16, 27], and many related tools like skytrace[2] and coinholmes[3] are available. Since BFS is insufficient to reveal the audit priority of different money tracking directions, the taint analysis technologies [9, 13, 18] have been proposed for bitcoin tracing according to the amount value and the order of multiple outputs in each transaction. However, these technologies rely on expert analysis and cannot automatically output the money flows from the source to the targets.

Based on the co-spending clustering heuristic in Bitcoin that the input addresses of the same transaction belong to the same entity [17], Huang et al. proposed to track the bitcoin flow of ransomware to when the money is being cashed out [11]. Meiklejohn et al. [12] utilized the change address heuristic to deanonymize the money flows in Bitcoin. Moreover, some rule-based transaction tracing methods were proposed for cross-chain scenarios and mixing scenarios. For example, Yousaf et al. traced the cross-chain money flows by identifying the transactions which happen close in time and have similar amount value [26]. However, these rule-based methods are designed for specific protocols and they are inapplicable to transaction tracing in account-based blockchains.

Personalized PageRank models the relevance of nodes in a network to a specific node, and has been widely used in web search

[2]https://www.certik.com/skytrace
[3]https://trace.coinholmes.com

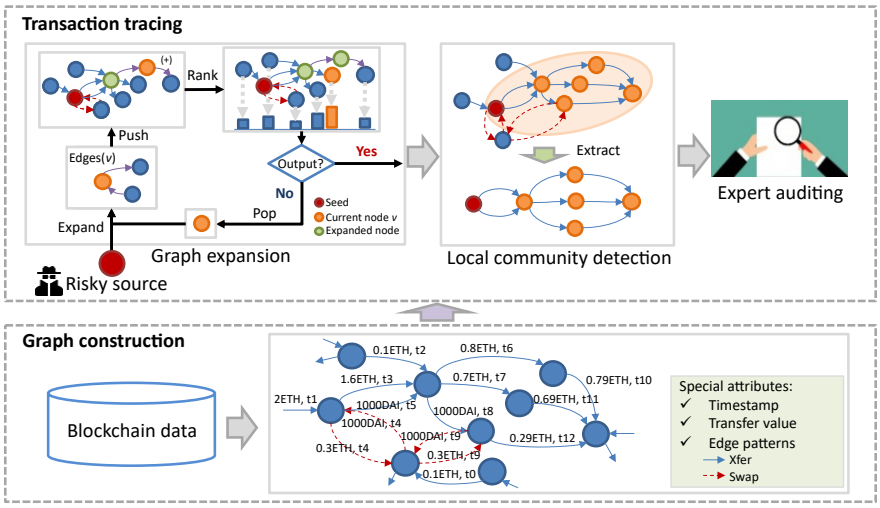


Figure 2: The framework of TRacer.

[15], recommendations [10], etc. Recently many variants and efficient approximations of personalized PageRank have been proposed for large-scale applications [7]. In this paper, we model the account-based blockchain data as a directed, weighted, temporal, and multi-relationship graph, and formalize the transaction tracing problem as a graph searching problem. We develop a scalable and intelligent transaction tracing tool for account-based blockchains based on personalized PageRank and its approximate solutions.

3 PRELIMINARIES

3.1 Account-based blockchains

Traditional Bitcoin-like blockchains are based on the transaction-centered model [21] whose building block is unspent transaction output (UTXO), which is an indivisible cryptocurrency chunk locked to a specific owner. Each transaction has multiple inputs made up of UTXOs and multiple outputs, and there is a change address in the outputs used to receive the change.

Different from UTXO-based blockchains, account-based blockchain systems like Ethereum and BSC have the concept of account similar to that of traditional banking accounts. There are two kinds of accounts in account-based blockchains, namely external owned account (EOA) and smart contract account. Accounts are the initiators of blockchain transactions and record some dynamic state information including account balance. Especially, each smart contract account is associated with a piece of executable bytecode. There are also two types of transactions in the systems. A transaction triggered by an EOA is called external transaction, while a transaction triggered by an invocation of the function in a smart contract account is called internal transaction. In addition, an EOA can invoke functions of a smart contract in an external transaction and further result in many internal transactions. Transaction hash consisting of a set of numbers and letters is used to uniquely identify a particular transaction from or to an EOA. Due to the support of smart contract, everyone can take the advantage of blockchain technology and build DApp projects in account-based blockchains. Besides the native currency in blockchain systems, there are many third-party tokens representing assets, currency, or access rights of projects in the account-based blockchain ecosystem. To facilitate token development and exchange, some token standards are launched in blockchain trading systems, e.g., the ERC20 token standard in Ethereum. There are also many DeFi DApps that offer financial services such as token lending and exchange.

3.2 Problem Definition

The transaction tracing task in account-based blockchain trading systems aims to trace the money flows from a given source to the targets that gather the money awaiting cashing out and point the priority money flows for auditors to further verify manually. By modeling the blockchain transaction relationships as graph where nodes indicate the accounts and edges indicate the money flow relationships, we can formulate the transaction tracing problem as follows.

Problem formulation. *Given a source node s in a transaction graph G, the goal is to search a connected money transfer subgraph G_s from s to its money flow targets around the neighborhood of s. G_s should contain as many target nodes as possible in the smallest possible size of graph for manual verification.*

3.3 Approximate Personalized PageRank

Personalized PageRank. The personalized PageRank vector $\boldsymbol{p}_s$ of a source node s in a graph $G = (V, E)$ is defined as the unique solution of Equation 1, i.e.,

$$\boldsymbol{p}_s = \alpha \boldsymbol{e}_s + (1 - \alpha)\boldsymbol{p}_s M, \tag{1}$$

where α is a teleport constant between 0 and 1, $\boldsymbol{e}_s$ is the indicator vector with a single nonzero element of 1 at s, M is a transition matrix and $M = D^{-1}A$ where D and A are degree matrix and adjacency matrix. The definition of personalized PageRank is equivalent to simulating a random walk starting from s, and $\boldsymbol{p}_s$ is a probability vector where $\boldsymbol{p}_s(u)$ is the probability that a certain random walk beginning at s terminates at u.

Approximate personalized PageRank (APPR). The first algorithm proposed to calculate personalized PageRank is power iteration [15], which requires high time complexity and not effective in large-scale networks. Therefore, various efficient approximate solutions of personalized PageRank have been proposed, and the most widely used one is called the "local push" algorithm [1]. This algorithm starts with all probability residual on the source node of the graph, and pushes the residual to the neighbors iteratively. During the iterations, the residual of each node can be transformed into the relevance to the source node. Finally an estimate of $\boldsymbol{p}_s$ can be obtained. In the algorithm, a residual vector $\boldsymbol{r}_s$ is used to maintain the residual, where $\boldsymbol{r}_s(u)$ denotes the residual of node u. By setting $\boldsymbol{p}_s = \vec{0}$ and $\boldsymbol{r}_s = \boldsymbol{e}_s$ for initialization, the local push procedure updates the value of $\boldsymbol{p}_s(u)$ as follows:

$$\begin{cases} \boldsymbol{p}_s(u) = \boldsymbol{p}_s(u) + \alpha \boldsymbol{r}_s(u) \\ r(v) = r(v) + (1-\alpha) r(u)/d(u), \end{cases}, \tag{2}$$

where $v \in N(u)$ is the neighbor of u, and $d(u)$ is the degree of u. The local push procedure stops when the residual of each node in G is within ϵ.

4 PROPOSED APPROACH

This section introduces the architecture, implementation, and theoretical analysis of TRacer.

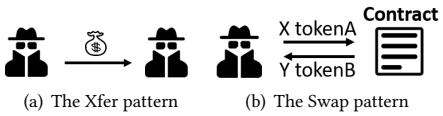
4.1 Architecture Overview

TRacer consists of three main modules including graph construction, graph expansion, and local community detection. Each module is described as follows:

- **Graph construction**: This module models the money transfer relationships between accounts as directed, weighted, temporal, and multi-relationship graphs.
- **Graph expansion**: Since the blockchain data contains billions of transactions which is too large for common graph algorithms, this module aims to find a relevant subgraph from a risky source. The module contains four operations: ***Expand*** collects all edges related to a given node, ***Push*** merges the collected edges to the subgraph, ***Rank*** computes the relevance of nodes in the subgraph to the source, and ***Pop*** selects a node for expanding. The graph expansion process terminates when the end condition is met.
- **Local community detection**: This module ***Extract***s a local community of the source node from the expended subgraph, in which nodes have higher relevance to the source than nodes out of the community.

4.2 Graph construction

Since there are multiple types of tokens in account-based blockchain trading systems, we formulate the money transfer relationships among accounts into a directed, weighted, temporal, and multi-relationship graph $G = (V, E)$, where V is the node set representing accounts, E is the edge set representing the token transfer relationships. An edge $e = (u, v, w, t, b, h)$ denotes that account u transfer w units token b to account v at timestamp t with a transaction hash h. We define mapping functions f_{src}, f_{tgt}, f_{amt}, f_{ts}, and f_{sym} to map each edge to its source, target, amount, timestamp, and token type respectively. There are multiple types of edges indicating the transfer relationships of different tokens.



(a) The Xfer pattern (b) The Swap pattern

Figure 3: (a) Xfer: Sending or receiving tokens. (b) Swap: Exchanging tokens for other kinds of tokens.

In addition, many blockchain services such as decentralized exchanges act as the intermediary for token swap. To reveal the token flows after users interact with these services, we categorize the money transfer relationships into two patterns, namely **Xfer** and **Swap**. As shown in Figure 3(a), accounts send or receive tokens through the Xfer pattern, and the related DeFi actions [24] in this pattern include: 1) ***transfer***: An account sends an amount of token to another account, 2) ***minting***: A token contract mints an amount of token to an account, and 3) ***burning***: An account burns an amount of token. While accounts exchange tokens for other kinds of tokens through the Swap pattern as shown in Figure 3(b), including three related DeFi actions: 1) ***add liquidity***: An account deposits an amount of token to a DeFi app, and receives a certain amount of Liquidity Provider (LP) token back, 2) ***remove liquidity***: An account sends an amount of LP token to a DeFi app for destroying and gets a certain amount of other tokens back, and 3) ***trade***: An account sells an amount of token A in a DeFi app for a certain amount of token B. We identify the transaction relationships involving both the sending of tokens and the receiving of another kind of tokens with the same hash as the relationships in the Swap pattern, and otherwise as the relationships in the Xfer pattern.

4.3 Graph Expansion

The graph expansion module aims to obtain a relevant subgraph expanded from a given seed via iterating four operations shown in Figure 2. In what follows, we introduce the design of these four operations.

4.3.1 Push and Rank: Transaction Tracing Rank. The *Push* operation merges the expanded edges to the subgraph in each iteration. While *Rank* calculates the relevance of nodes in the subgraph to the source.

We introduce APPR to calculate the node relevance. More specifically, the Rank operation executes the local push procedure for the incremental update of node relevance. Based on the characteristics of blockchain transaction graph in account-based blockchains, we propose a novel local push algorithm named **Transaction Tracing Rank (TTR)** to compute the node relevance in blockchain transaction graph. We develop four local push strategies in TTR, namely **tracing tendency, weighted pollution, temporal reasoning, and token redirection**. We use $r_s(u, t, b)$ to denote the residual of node u brought from token $b \in B$ at timestamp $t \in \Gamma$, where Γ is the timestamp set and B is the token symbol set in the subgraph. In this way, the local push procedure in TTR is described as Algorithm 1. During initialization, a part of the residual of node u is transformed into the relevance (line 1). Moreover, other parts of the residual are pushed to the neighbors according to the four strategies.

Algorithm 1 TTR local push

Input: Node u, edges $E(u)$ related to u, edge mapping functions f_{src}, f_{tgt}, f_{amt}, f_{ts}, f_{sym}, rank $\boldsymbol{p}_s$, residual r_s, timestamps Γ, token symbols B, teleport constant α, and tracing tendency β.
Output: The updated $\boldsymbol{p}_s$ and r_s.

1: $\boldsymbol{p}_s(u) = \boldsymbol{p}_s(u) + \alpha \sum_{b\in B}\sum_{t\in\Gamma} r_s(u,t,b)$
2: $r'_s(u,t,b) = r_s(u,t,b), r_s(u,t,b) = 0, \forall t \in \Gamma, \forall b \in B.$
3: **for** $(t,b) \in \Gamma \times B$ **do**
4: $E_{out} = \{e \in E(u) | f_{ts}(e) > t \wedge f_{src}(e) = u \wedge f_{sym}(e) = b\}$
5: $E_{in} = \{e \in E(u) | f_{ts}(e) < t \wedge f_{tgt}(e) = u \wedge f_{sym}(e) = b\}$
6: **for** $E' \in \{E_{out}, E_{in}\}$ **do**
7: $\gamma = \beta$ *if* $E' == E_{out}$ *else* $(1-\beta)$
8: **for** $e \in E'$ **do**
9: $E'_\rho = \rho(\{e\}, E(u))$ // obtained by Equation 3
10: **for** $e' \in E'_\rho$ **do**
11: $\Delta = \frac{(1-\alpha)\gamma f_{amt}(e)}{|E'_\rho| \sum_{e''\in E'} f_{amt}(e'')} r'_s(u,t,b)$
12: $v = f_{src}(e')$ *if* $E' == E_{in}$ *else* $f_{tgt}(e')$
13: $r_s(v, f_{ts}(e'), f_{sym}(e'))+ = \Delta$
14: **end for**
15: **end for**
16: $r_s(u,t,b) = (1-\alpha)\gamma r'_s(u,t,b)$ if $|E'| == 0$.
17: **end for**
18: **end for**
19: **return** $\boldsymbol{p}_s, r_s$

Tracing tendency. Since a money transfer relationship between accounts is directed, the attention to in-degree neighbors and out-degree neighbors during the local push procedure can be different. In most cases, tracing for the destinations of money flows oriented from a particular source needs to pay more attention to the out-degree neighbors. Therefore, we define an attention coefficient $0 \leq \beta \leq 1$. During the residual propagation, the out-degree neighbors in a transaction relationship can get a higher residual when $\beta > 0.5$, and the in-degree neighbors can receive a higher residual when $\beta < 0.5$. As Figure 4(a) shows, the in-degree neighbors and the out-degree neighbors obtain a different weight in this strategy. Line 6-17 in Algorithm 1 describe the residual propagation procedure of different directions with the attention coefficient assigned at line 7.

Weighted pollution. In blockchain transaction graph, the strength of money transfer relationships is weighted by the transaction amount. For each node, a neighbor trading a larger amount of money with this node is considered to be more relevant to the node. Therefore, in this strategy we take the weight of the money transfer relationships into account. Figure 4(b) shows an example of this strategy that neighbors of u associated by edges with greater weight can obtain more residual during the propagation. In this way, the residual of a node is distributed to each neighbor associated by edge $e \in E'$ according to the ratio of edge weight $f_{amt}(e)/\sum_{e''\in E'} f_{amt}(e'')$ as shown in Algorithm 1 line 11, where E' is the set of edges.

Temporal reasoning. Blockchain transactions are recorded in blocks chronologically. Each block contains a specific timestamp. With the timestamp information, tracing for the targets of money flows usually follows the paths with increasing timestamps, while tracing back to the source of money flows follows the paths with decreasing timestamps. Therefore, in this strategy we take the temporal information into consideration. For example in Figure 4(c), the residual from the input transaction at $t2$ is pushed out through the outgoing edges after $t2$ and the incoming edge before $t2$. Algorithm 1 line 4-5 show the edge set for residual propagation that satisfies this strategy. Moreover, the residual is pushed to the node itself if the edge set is empty in Algorithm 1 line 16, indicating that the funds have not been transferred from the node.

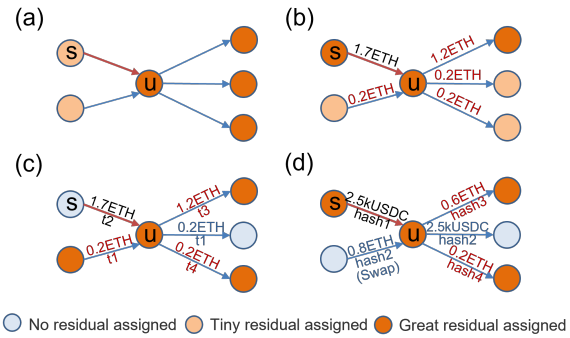


Figure 4: (a) Tracing tendency. Different attention is assigned to the out-degree neighbors and in-degree neighbors. (b) Weighted pollution. The higher the edge weight, the closer the relationship. (c) Temporal reasoning. Tracing money flow in chronological order, and $t1 < t2 < t3 < t4$ in this example. (d) Token redirection. Uncovering the token flows even though there exist complex DeFi actions in the Swap pattern.

Token redirection. This strategy makes the effort to uncover the flow of interesting tokens based on the transaction patterns of Xfer and Swap. As Figure 4(d) shows, the residual of node u brought from the USDC token in an Xfer edge with hash1 should be pushed through the edges with hash3 and hash4, rather than the USDC outgoing edge with hash2. Since the edges with hash2 are in a Swap pattern and swapped the USDC token into ETH. To achieve the redirection of token flows in complex transaction actions, we define a recursive function $\rho(\cdot,\cdot)$ which can find the initial state of a set of incoming edges before the Swap operations and the final state of a set of outgoing edges after the Swap operations within a node. For example, the initial state before Swap of the incoming edge with hash2 in Figure 4(d) is the incoming edge with hash1, and the final state after Swap of the outgoing edge with hash2 is the outgoing edges with hash3 and hash4. More specifically, as shown in Algorithm 1 line 8-14, the residual of a specific token should be pushed through the redirected edges. Therefore, $\rho(\cdot,\cdot)$ satisfies the following recursive equation:

$$\rho(\mathcal{E}, E(u)) = \mathcal{E}_{xfer} \cup \rho(\bigcup_{e\in\mathcal{E}_{swap}} redirect(e), E(u)), \quad (3)$$

where $\mathcal{E}$ is a set of edges for redirection, $\mathcal{E}_{xfer} \subset \mathcal{E}$ is a set of Xfer edges, $\mathcal{E}_{swap} \subset \mathcal{E}$ is a set of Swap edges, and $redirect(\cdot)$

Algorithm 2 TTR-based Local Community Detection

Input: The source node s, the subgraph of graph expansion $G_s = (V_s, E_s)$, and the TTR score vector $\boldsymbol{p}_s$.
Output: The local community with nodes set S.

```
S = {s}
S̄ = V_s \ S
while Φ(S) ≥ φ do
    u = arg max_{v∈S̄} p_s(v)
    S = S ∪ {u}
    S̄ = S̄ \ {u}
end while
return G_s.subgraph(S)
```

selects the edges in the token types before/after Swap for incoming/outgoing edges $e \in \mathcal{E}_{swap}$ from edges $E(u)$ related to u. Especially, $\rho(\emptyset, E(u)) = \emptyset$.

4.3.2 Pop and Expand: Greedy Selection. The *Pop* operation selects a node from the subgraph for the next round expansion iteration, and the *Expand* operation expands from a node by collecting all the related edges of this node. Note that the graph expansion in TRacer terminates when the residual of all nodes in the subgraph is below a threshold $\epsilon \in (0, 1)$, i.e.,

$$\max_{u \in V}\Big(\sum_{t \in \Gamma}\sum_{b \in B} r_s(u, t, b)\Big) < \epsilon. \tag{4}$$

Thus we proposed the **greedy selection** for the *Pop* operation to achieve the condition in Equation 4, i.e., the *Pop* operation selects the node in the subgraph with the highest residual.

4.4 Local Community Detection

Referring to Figure 2, the *Extract* operation employs local community detection to construct a small-scale local community from the expanded graph. For a particular risky source node, the importance rank of the nodes within the local community is significantly higher than the external nodes, making it easy for further expert auditing.

The less conductance [1, 2] of the local community, the higher rank of the nodes in the local community than external nodes, in which the conductance is:

$$\Phi(S) = \frac{\boldsymbol{p}_s(\partial(S))}{\boldsymbol{p}_s(S)}, \tag{5}$$

where S denotes the nodes of the local community, the boundary $\partial(S) = \{v | (u, v) \in E \wedge u \in S \wedge v \in \bar{S}\}$, $\bar{S}$ is the complement of S and $\boldsymbol{p}_s(S) = \sum_{u \in S} \boldsymbol{p}_s(u)$ denotes the sum of rank over each node in the local community. Given a specific threshold $\varphi > 0$ for conductance, the local community detection finds the local community satisfying:

$$\Phi(S) < \varphi. \tag{6}$$

With $S = \{s\}$ as the initialization, Algorithm 2 describes how to find the local community satisfied the Equation 6.

4.5 Theoretical properties

In this part, we discuss the theoretical properties of TRacer, and prove that our method is able to finish the transaction tracing task in large-scale transaction graphs with a constant time cost. Besides, we discuss the upper limit of the tracing depth in TRacer.

As the description in Proposition 4.1, the cost of TRacer is independent of the graph size, indicating that TRacer is able to trace on the large-scale transaction graph with a low cost.

Proposition 4.1. *The iteration of graph expansion runs $O(\frac{1}{\epsilon\alpha})$ times, and the number of nodes with non-zero values in the output TTR score is at most $O(\frac{1}{\epsilon\alpha})$, which guarantees the cost of local community detection is $O(\frac{1}{\epsilon\alpha})$.*

Proof. This follows from Andersen et al. [1], Lemma 2. □

In addition, what depth can TRacer trace in a transaction graph from the source node is described in Proposition 4.2.

Proposition 4.2. *An n-hop neighbor of the source node can be found in the graph obtained by graph expansion, in which n satisfies:*

$$n \le \frac{log(\epsilon)}{log(1-\alpha)} + 1. \tag{7}$$

Proof. In order to maximize the rank of the neighbors far away from the source node, α needs to be as small as possible, and β needs to be as close as possible to 0 or 1, which ensures that the residual can be pushed to a specific direction. Let the sum of residual pushed from the source node s to the n-hop neighbors be $r^{(n)}$. Considering $\beta = 1$ here, $r^{(n)}$ can be obtained by:

$$\begin{cases} r^{(1)} = (1-\alpha), \\ r^{(2)} \ge (1-\alpha)(r^{(1)} - k_1\epsilon) = (1-\alpha)^2 - (1-\alpha)k_1\epsilon, \\ \cdots\cdots \\ r^{(n)} \ge (1-\alpha)(r^{(n-1)} - k_{n-1}\epsilon) \\ \qquad = (1-\alpha)^n - \sum_{i=1}^{n-1}(1-\alpha)^{n-i}k_{n-i}\epsilon, \end{cases} \tag{8}$$

where k_i represents the number of nodes with residual less than ϵ in the i-order neighbors of the source node. Therefore, $r^{(n)}$ obtains the maximum value when:

$$\sum_{i=1}^{n-1}(1-\alpha)^{n-i}k_{n-i}\epsilon = 0, \tag{9}$$

which means the residual of each i-order node is greater or equal than ϵ. This situation exists when the source node is in a path-like graph, whose adjacency matrix A satisfies $A(i, i+1) = 1$ and other elements are 0. Considering the end condition of the local push procedure, the residual of n-hop neighbors is:

$$\begin{cases} (1-\alpha)^n < \epsilon \\ (1-\alpha)^{n-1} \ge \epsilon \end{cases} \Rightarrow \frac{log(\epsilon)}{log(1-\alpha)} < n \le \frac{log(\epsilon)}{log(1-\alpha)} + 1. \tag{10}$$

□

5 EXPERIMENTS

In this section, we conduct experiments to evaluate the effectiveness of TRacer on a large-scale real-world dataset. Besides, we conduct case studies with network visualization techniques.

Table 1: Statistics of the transaction record data related to the cases

Field	Description	Number
Source nodes	The source node related to this case, such as the hacker account and the scam contract.	20
Target nodes	A set of target nodes related to this case, such as exchange wallets and mixing services.	0.87K
Blocks	The blocks related to these cases.	23.5M
Transactions	The transactions contained in the blocks related to these cases.	4.83B

5.1 Experimental Setups

5.1.1 Dataset. We contribute a benchmark dataset including 20 transaction tracing cases in the recent 5 years across three account-based blockchains, i.e., Ethereum, Binance Smart Chain, and Polygon. These cases are initialized by various illegal activities containing hacker attacks, Rug-pull, and scams which have caused billions of dollars in losses. All these cases are reported by blockchain security companies and verified by experts coming from Certik [4], Peckchield [5], Chainalysis [6] and so on. Some statistics of this dataset are shown in Table 1. Note that the transaction data in the dataset is obtained from the open APIs[7]. As we can see, the activities of these cases have acrossed millions of blocks, and we have to trace the money flows of the sources among more than 4 billion transactions.

5.1.2 Compared Methods. We compare our method with several baseline blockchain transaction tracing methods. For a fair comparison, we use the general framework of TRacer to reproduce the following comparison methods, including:

- **BFS** [27]: Breadth-First Search, which is the first and the most commonly used transaction tracing method.
- **Poison** [13]: A kind of taint analysis technology in blockchain transaction tracing. Each output of a transaction with a dirty input is considered to be tainted in this method.
- **Haircut** [13]: A kind of taint analysis technology in blockchain transaction tracing. Each output of a transaction with a dirty input is considered to be tainted partially according to the amount value in this method.
- **APPR** [1]: The approximate personalized PageRank algorithms, which can calculate the relevance of nodes in a network to a given source node with an extremely low cost.

The details of implementing the above transaction tracing technologies can be found in our GitHub page [8].

5.1.3 Experimental Settings. Since the increase of depth can lead to the exponential growth of the size of output graph in BFS and Poison, bringing great difficulty to transaction auditing. During our experiments, we set the upper limit of the tracing depth in these two methods is 2. In addition, we use the Haircut method to trace the "dirty money" from the source node until the amount proportion of "dirty money" of all nodes is less than 0.1% of that from the source node. Moreover, we set $\alpha = 0.15$, $\epsilon = 10^{-3}$ for APPR and TTR, and $\varphi = 10^{-3}$, $\beta = 0.7$ for TTR to ensure that our method is able to find the paths among the source node and the target nodes with 42-hop at most, according to Proposition 4.2.

[4]https://www.certik.com/
[5]https://peckshield.com/
[6]https://www.chainalysis.com/
[7]https://blockscan.com
[8]https://github.com/wuzhy1ng/BlockchainSpider

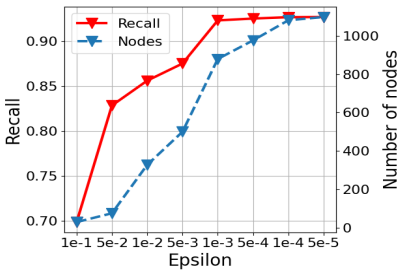


Figure 5: The relationship among Epsilon, Recall, and Node number. Note that the recall has reached 70% with $\epsilon = 10^{-1}$ merely, and the increment of recall becomes slow when ϵ is less than 10^{-3}.

5.1.4 Metrics. we report the average of the following metrics in all cases to measure the effectiveness of transaction tracing:

- **Recall**: The recall evaluates how many target nodes can be traced by a method, which is defined as: $Recall = \frac{|V_t|}{|\bar{V}_t|}$, where $|V_t|$ is the number of traced target nodes and $|\bar{V}_t|$ is the number of all target nodes in a case.
- **Number of nodes**: This metric measures the number of nodes in the output graph of a case. A smaller output graph with recall ensured is easier for expert auditing.
- **Tracing depth**: This metric measures how deep can a transaction tracing method traverse the transaction graph from a source node, indicating that up to K-hop neighbors of the source node are detected.

5.2 Experimental Results

5.2.1 Scalability vs. Performance. The approximation parameter ϵ is an important hyper-parameters in modulating the scalability. To examine the effect of ϵ on the performance, we repeat experiments with different values of ϵ and report the recall as well as the number of traced nodes. As Figure 5 shows, the recall has reached 70% when $\epsilon = 10^{-1}$. When ϵ is less than 10^{-3}, the increment of recall becomes slow, and the number of nodes rapidly increases. Therefore, setting $\epsilon = 10^{-3}$ can guarantee a higher recall and fewer nodes with a relatively low cost in experiments, and we set $\epsilon = 10^{-3}$ for TRacer.

5.2.2 Comparative experiment. Table 2 shows the performance of different methods, from which we can obtain the following observations. ***Firstly***, the output graphs of BFS and Poison contain an extremely large number of nodes, which brings great difficulty to transaction auditing even through detecting more than 70% target nodes. ***Secondly***, the output graph of Haircut has fewer nodes than BFS and Poison with a greater tracing depth, but the recall is too low to achieve effective tracing. ***Thirdly***, fewest nodes are obtained by APPR, ensuring the detection results can be easier audited by

Table 2: Performance comparison between baselines and TRacer

Methods	Recall (%)	Number of nodes (K)	Tracing depth
BFS	77.02	52.50	2.00
Poison	70.06	41.45	2.00
Haircut	58.85	10.35	4.15
APPR	71.92	**0.66**	3.60
TRacer	**92.31**	0.87	**5.05**

Table 3: Ablation experiment.

Graph construction with DeFi patterns	Graph expansion	Local community detection	Recall (%)	Number of nodes (K)
√	√	√	92.3	0.87
	√	√	80.3	0.55
√	√		95.9	57.5

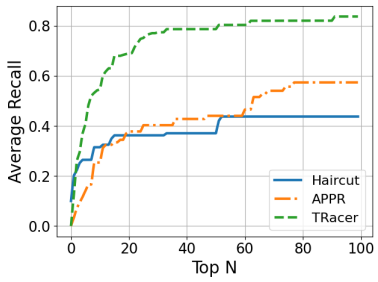


Figure 6: Top n most relevant nodes and the recall.

experts. **Fourthly**, TRacer obtains better performance than APPR. On the basis of the advantages of APPR, the recall and tracing depth of TRacer are significantly better than other methods.

5.2.3 Ablation experiment. TRacer is consist of three modules including graph construction, graph expansion, and local community detection. In order to discuss the function of different modules, we conduct an ablation study and report the performance of TRacer in Table 3 after removing the DeFi pattern recognition in graph construction and the local community detection. When the DeFi pattern recognition is removed in the graph construction module, the recall decreases significantly, which shows that the understanding of DeFi patterns in TRacer can help trace the money flows effectively. Moreover, if the local community detection module is removed, the number of nodes increases significantly with the weakly improvement of recall, which shows that the local community detection module can find the nodes strongly associated with the source node at the cost of a small recall loss.

5.2.4 TopN Recall. Since the rank of a node represents the relevance relationship between this node and the source node, we can audit the nodes according to the descending order of rank. To compare the rank-based methods including Haircut, APPR, and TRacer, we take out the top n most relevant nodes to the source

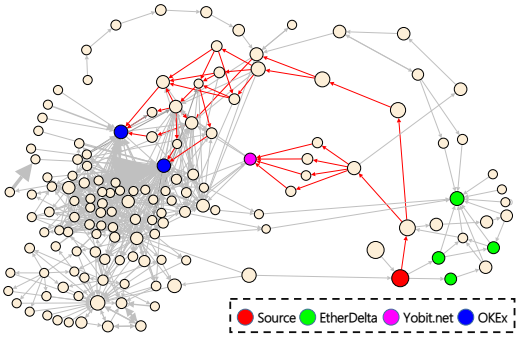


Figure 7: Tracing visualization for Cryptopia. Besides the EtherDelta exchange marked by experts, our method also finds the other target exchanges including Yobit.net and OKEx.

and calculate the recall for different n. The result is displayed in Figure 6, where the curve of TRacer shows a better performance than other methods. In addition, TRacer achieves a 25% recall gain over APPR when $n > 50$.

5.3 Case Study

In this part, we visualize the traced money flow of two cases with Gephi 0.9.2 [4], in order to evaluate the feasibility of TRacer.

5.3.1 Cryptopia. The Cryptopia exchange was attacked by hackers in May 2019. According to the tracing result published by CoinHolmes[9], the source node with prefix 0xd4e79 [10] possessed 30.8K stolen ETH from Cryptopia and transferred about 10K ETH to 4 target nodes labeled as EtherDelta.

Figure 7 presents the transaction tracing result of our method, where the source node and the target nodes are marked with labels. Additionally, the node size is proportional to its rank score for each node, so the higher the rank is, the larger the node diameter is. Based on the tracing result, we can find 4 target nodes labeled as EtherDelta (an exchange) in the 2-hop neighborhood of the source node easily, which is consistent with the results in CoinHolmes.

Moreover, another 4-hop neighbor of the source node labeled as Yobit.net[11] and two nodes labeled as OKEx can be found in the figure, which is not reported by CoinHolmes. In fact, Yobit.net and OKEx are exchanges enabling the hacker to cash out the stolen ETH. According to the traced money flows in the figure, about 1420 stolen ETH is transferred into Yobit.net, and 18.47K stolen ETH is transferred into OKEx. Therefore, more than 97% of the stolen ETH of Cryptopia are traced by our method in this case.

5.3.2 Kucoin. The Kucoin exchange was attacked by hackers in September 2020, and there is still a large number of stolen funds that have not been traced until now. As shown in Figure 8, the source node [12] of the hackers obtained ETH from the nodes labeled as Kucoin, and then transferred most of the stolen ETH to a famous privacy-preserving protocol named Tornado Cash [3].

[9] https://trace.coinholmes.com
[10] https://etherscan.io/address/0xd4e79226f1e5a7a28abb58f4704e53cd364e8d11
[11] https://etherscan.io/address/0xf5bec430576ff1b82e44ddb5a1c93f6f9d0884f3
[12] https://etherscan.io/address/0xeb31973e0febf3e3d7058234a5ebbae1ab4b8c23

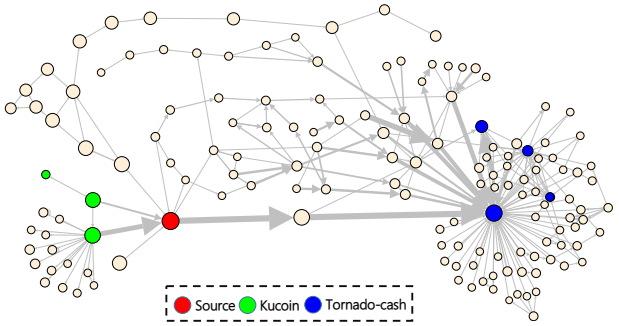


Figure 8: Tracing visualization for Kucoin. A large number of ETH was transferred to Tornado Cash.

Through the tracing result of our method, a total amount of 13.8K ETH was transferred to the 100 ETH pool of Tornado Cash, which is similar to the conclusion of experts in SlowMist[13]. As Tornado Cash is a decentralized mixing service, it is impossible for us to obtain valuable KYC information from this project to identify the hackers. In addition, since Tornado Cash is designed based on zero-knowledge proof, liquidity mining, and smart contract, it is difficult to trace the downstream fund flow when money is transferred into Tornado Cash. Therefore, some researchers have conducted analysis on Tornado Cash in recent years [5].

6 CONCLUSION

In this paper, we studied the transaction tracing problem on account-based blockchain trading systems and proposed the first intelligent transaction tracing tool named TRacer. Compared with existing methods based on heuristics and taint analysis, which usually requires expert experience and manual intervention, TRacer show obvious superiority in terms of universality, effectiveness and time cost. In TRacer, we first formulated the transaction records in each account-based blockchain as a directed, weighted, temporal, and multi-relationship graph, which is able to represent the rich semantics of complex multi-token transaction relationships in the system. Then we proposed to use personalized PageRank-based graph searching technologies on this complex graph to trace the money flows. Specifically, we introduced novel approximate personalized PageRank strategies to realize effective and low-cost transaction tracing, which can also handle the complex DeFi transaction actions in account-based blockchains. The theoretical analysis and experimental results demonstrated the effectiveness of TRacer. In the future, we will further delve into blockchain transaction tracing by integrating more transaction features like bytecodes and logs and design effective methods for blockchain systems with privacy-enhancing mechanisms.

REFERENCES

[1] Reid Andersen, Fan Chung, and Kevin Lang. 2006. Local graph partitioning using pagerank vectors. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society, Berkeley, California, USA, 475–486.
[2] Reid Andersen, Fan Chung, and Kevin Lang. 2007. Local partitioning for directed graphs using PageRank. In *Proceedings of the International Workshop on Algorithms and Models for the Web-Graph*. Springer, San Diego, CA, USA, 166–178.
[3] ayefda. 2021. Introduction of Tornado.Cash. https://docs.tornado.cash/.
[4] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. 2009. Gephi: An open source software for exploring and manipulating networks. In *Proceedings of the Third International Conference on Weblogs and Social Media*. AAAI Press, San Jose, California, USA, 361–362.
[5] Ferenc Béres, István András Seres, András A Benczúr, and Mikerah Quintyne-Collins. 2020. Blockchain is watching you: Profiling and deanonymizing Ethereum users. *arXiv preprint arXiv:2005.14051* (2020).
[6] Binance. 2020. Binance chain documentation. https://docs.binance.org/.
[7] Aleksandar Bojchevski, Johannes Klicpera, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. 2020. Scaling graph neural networks with approximate pagerank. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, CA, USA, 2464–2473.
[8] Chainalysis. 2022. Crypto Crime Trends for 2022: Illicit Transaction Activity Reaches All-Time High in Value, All-Time Low in Share of All Cryptocurrency Activity. https://blog.chainalysis.com/reports/2022-crypto-crime-report-introduction/.
[9] Giuseppe Di Battista, Valentino Di Donato, Maurizio Patrignani, Maurizio Pizzonia, Vincenzo Roselli, and Roberto Tamassia. 2015. Bitconeview: Visualization of flows in the Bitcoin transaction graph. In *Proceedings of 2015 IEEE Symposium on Visualization for Cyber Security*. IEEE Computer Society, Chicago, IL, USA, 1–8.
[10] Pankaj Gupta, Ashish Goel, Jimmy Lin, Aneesh Sharma, Dong Wang, and Reza Zadeh. 2013. WTF: The Who to Follow Service at Twitter. In *Proceedings of the 22nd International Conference on World Wide Web*. Association for Computing Machinery, Rio de Janeiro, Brazil, 505–514.
[11] Danny Yuxing Huang, Maxwell Matthaios Aliapoulios, Vector Guo Li, Luca Invernizzi, Elie Bursztein, Kylie McRoberts, Jonathan Levin, Kirill Levchenko, Alex C. Snoeren, and Damon McCoy. 2018. Tracking Ransomware End-to-end. In *Proceedings of the 2018 IEEE Symposium on Security and Privacy*. IEEE Computer Society, San Francisco, California, USA, 618–631.
[12] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M Voelker, and Stefan Savage. 2013. A fistful of Bitcoins: Characterizing payments among men with no names. In *Proceedings of the 2013 conference on Internet measurement conference*. ACM press, Barcelona, Spain, 127–140.
[13] Malte Möser, Rainer Böhme, and Dominic Breuker. 2014. Towards risk scoring of Bitcoin transactions. In *Proceedings of the International Conference on Financial Cryptography and Data Security*. Springer, Barbados, 16–32.
[14] Frédérique Oggier, Anwitaman Datta, and Silivanxay Phetsouvanh. 2020. An ego network analysis of sextortionists. *Social Network Analysis and Mining* 10, 1 (2020), 44.
[15] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web*. Technical Report. Stanford InfoLab.
[16] Silivanxay Phetsouvanh, Frédérique Oggier, and Anwitaman Datta. 2018. Egret: Extortion graph exploration techniques in the Bitcoin network. In *Proceedings of IEEE International Conference on Data Mining Workshops*. IEEE, Singapore, 244–251.
[17] Fergal Reid and Martin Harrigan. 2013. *An analysis of anonymity in the Bitcoin system*. Springer, 197–223.
[18] Tin Tironsakkul, Manuel Maarek, Andrea Eross, and Mike Just. 2019. Probing the mystery of cryptocurrency theft: An investigation into methods for cryptocurrency tainting analysis. *arXiv preprint arXiv:1906.05754* (2019).
[19] Mark Weber, Giacomo Domeniconi, Jie Chen, Daniel Karl I Weidele, Claudio Bellei, Tom Robinson, and Charles E Leiserson. 2019. Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. *arXiv preprint arXiv:1908.02591* (2019).
[20] Jiajing Wu, Jieli Liu, Weili Chen, Huawei Huang, Zibin Zheng, and Yan Zhang. 2021. Detecting mixing services via mining Bitcoin transaction network with hybrid motifs. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* (2021). to be published, doi: 10.1109/TSMC.2021.3049278.
[21] Jiajing Wu, Jieli Liu, Yijing Zhao, and Zibin Zheng. 2021. Analysis of cryptocurrency transactions from a network perspective: An overview. *Network and Computer Applications* 190 (2021), 103139.
[22] Jiajing Wu, Qi Yuan, Dan Lin, Wei You, Weili Chen, Chuan Chen, and Zibin Zheng. 2022. Who Are the Phishers? Phishing Scam Detection on Ethereum via Network Embedding. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 52, 2 (2022), 1156–1166.
[23] Lei Wu, Yufeng Hu, Yajin Zhou, Haoyu Wang, Xiapu Luo, Zhi Wang, Fan Zhang, and Kui Ren. 2021. Towards understanding and demystifying Bitcoin mixing services. In *Proceedings of the Web Conference 2021*. Association for Computing Machinery, New York, NY, USA, 33–44.
[24] Siwei Wu, Dabao Wang, Jianting He, Yajin Zhou, Lei Wu, Xingliang Yuan, Qinming He, and Kui Ren. 2021. DeFiRanger: Detecting Price Manipulation Attacks on DeFi Applications. *arXiv preprint arXiv:2104.15068* (2021).
[25] Hao Yin, Austin R Benson, Jure Leskovec, and David F Gleich. 2017. Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD International*

[13] https://coinyuppie.com/uncovering-tornado-cashs-anonymity/

Conference on Knowledge Discovery and Data Mining. ACM, Halifax, NS, Canada, 555–564.

[26] Haaroon Yousaf, George Kappos, and Sarah Meiklejohn. 2019. Tracing transactions across cryptocurrency ledgers. In *Proceedings of the 2019 USENIX Security Symposium*. USENIX Association, Santa Clara, CA, USA, 837–850.

[27] Chen Zhao and Yong Guan. 2015. A graph-based investigation of Bitcoin transactions. In *Proceedings of IFIP International Conference on Digital Forensics*. Springer International Publishing, Orlando, FL, USA, 79–95.

[28] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Xiangping Chen, and Huaimin Wang. 2018. Blockchain challenges and opportunities: A survey. *International Journal of Web and Grid Services* 14, 4 (2018), 352–375.