Transaction Tracking on Blockchain Trading Systems using Personalized PageRank

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

Due to the pseudonymous nature of blockchain, various cryptocurrency systems like Bitcoin and Ethereum have become a hotbed for illegal transaction activities. The ill-gotten profits on many blockchain trading systems are usually laundered into concealed and "clean" fund before being cashed out. Recently, in order to recover the stolen fund of users and reveal the real-world entities behind the transactions, much effort has been devoted to tracking the flow of funds involved in illegal transactions. However, current approaches are facing difficulty in estimating the cost and quantifying the effectiveness of transaction tracking. This paper models the transaction records on blockchain as a transaction network, tackle the transaction tracking task as graph searching the transaction network and proposes a general transaction tracking model named as Push-Pop model. Using the three kinds of heuristic designs, namely, tracking tendency, weight pollution, and temporal reasoning, we rewrite the local push procedure of personalized PageRank for the proposed method and name this new ranking method as Transaction Tracking Rank (TTR) which is proved to have a constant computational cost. The proposed TTR algorithm is employed in the Push-Pop model for efficient transaction tracking. Finally, we define a series of metrics to evaluate the effectiveness of the transaction tracking model. Theoretical and experimental results on realist Ethereum cases show that our method can track the fund flow from the source node more effectively than baseline methods.

CCS CONCEPTS

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

KEYWORDS

Blockchain, Transaction tracking, Personalized PageRank, Local community discovery

*Both authors contributed equally to this research.
†Both authors contributed equally to this research.

Conference'17, July 2017, 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. Transaction Tracking on Blockchain Trading Systems using Personalized PageRank. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 10 pages. https://doi.org/xx.xxxx/xxxxxxx.xxxxxxx

1 INTRODUCTION

Since the debut of Bitcoin in 2009 [14], various cryptocurrencies and blockchain technology which provide decentralized environments for cryptocurrency transactions have gained increasing popularity and attention in recent years [24]. Incorporating peer-to-peer (P2P) technology, cryptography, and consensus protocol, blochchain is regarded as the fundamental technology supporting cryptocurrencies and allows users to participate in cryptocurrency trading with pseudonyms [31].

However, the pseudonymous nature of Blockchain trading systems has also attracted a variety of illegal transaction activities like scams, extortion, and hackers. For example, a vulnerability of "The DAO" smart contract in Ethereum [22] was exploited by hackers in 2016, and a large amount of investment worth over $70 million was stolen [8]. According to a report given by Chainalysis [9], a blockchain data analysis service provider, cryptocurrency transactions with a total amount of more than $11.5 billion worth of cryptocurrency transactions were associated with illegal transaction activities. Moreover, according to a recent report of Peckshield [18], a blockchain security company, during the first half of 2021, the losses caused by illegal transaction activities in the cryptocurrency related industry have exceeded $14.24 billion. Due to the enforcement of the know-your-customer (KYC) process in some blockchain exchanges, illegal profits on many blockchain trading systems are usually laundered into concealed and "clean" fund flows before being cashed out. Therefore, in order to crack down on illegal transaction activities on blockchain, a wealth of efforts from both academia and industry have been devoted to tracking the flow of funds involved in illegal transaction activities [10, 25, 28], the purpose being to understand real-world entities behind each transaction, help victims recover the losses, and thus make it easier to counter money laundering, fraud, dark web trading, and other digital asset crimes.

Generally speaking, transaction tracking tasks running on a constructed blockchain transaction network aim at finding the paths among the source node and the target nodes. Here, the source node represents a fund tracking object such as a scam deployer decamped with a large amount of illegal gains, and the target nodes represent the blockchain addresses used to gather the "clean" funds awaiting

cashing out. Usually, the target nodes are addresses possessing the laundered funds or exchange deposit addresses. After locating these targets, the illegal funds can be intercepted and the relative criminals can be identified offline according to the identity information provided by exchanges [23].

Yet transaction tracking on blockchain trading systems is a rather challenging task. ***Firstly***, because of the pseudonymous nature of blockchain, it is unlikely to enforce the KYC process to verify the identities and ascertain the potential risks of users during cryptocurrency transactions [24], which makes it extremely difficult to determine the flow direction of a certain amount of money. ***Secondly***, without identity information, conducting money tracking in a large amount of blockchain transaction records is like finding a needle in the haystack, which requires a low computational cost of the transaction tracking.

Current approaches [16, 20, 30] for blockchain transaction tracking are mainly based on the Bitcoin system and inspired by graph searching [1, 26] and taint analysis [13, 21] technologies. The main idea of existing studies is to start from the source node and search the possible paths of funds between the source and target nodes via certain rules. However, as a newly emerging area, there still exist some limitations in much of the existing work: **(1)** Without theoretical proofs, the end condition of most of the existing methods relies on experts, which makes it difficult to estimate the computational cost of these methods. Therefore, the time cost of running existing transaction tracking methods on a large-scale transaction network may be extremely high when the end condition is not well designed. **(2)** Most of the existing studies are heuristic methods designed for some specific cases, which rely heavily on expert experience. Currently, there is no common framework and evaluation criteria for transaction tracking on blockchain. Therefore, it is difficult to evaluate the universality and superiority of the existing methods.

In order to solve the above-mentioned problems, we propose a general framework called Push-Pop model for transaction tracking on blockchain trading systems. Given a source node, the framework can automatically find the paths of fund flows oriented from the source node. Then, we design the **Transaction Tracking Rank** (TTR) algorithm for the Push-Pop model based on Personalized PageRank methods, where Personalized PageRank is usually used to calculate node importance in a network from a point of view of a given node [17], ensuring the fund flow paths of the source node and the important nodes in the paths can be found. For the challenge brought by the pseudonymous nature, we utilize heuristic knowledge to design the TTR algorithm, so that the target nodes can obtain greater importance for identifying easier. For the problem of cost, we introduce the Approximately Personalized PageRank (APPR) [2, 3, 11, 27] algorithm to obtain the approximate solution of Personalized PageRank. It is worth mentioning that our proposed method has been applied and evaluated in transaction tracking tasks on several blockchain trading systems including Bitcoin, Ethereum, and Binance Smart Chain [7] and so on, but may not be suitable for some privacy-enhancing blockchain trading systems like the Monero [15]. Finally, we evaluate our method with metrics and network visualization analysis on five Ethereum transaction tracking cases. The main contribution of this work can be summarized as follows:

- **Problem**: We give a problem definition of transaction tracking on Blockchain trading systems as finding the paths among the source node and the target nodes, and propose a general transaction tracking model named Push-Pop model.
- **Approach**: We propose the TTR algorithm running on a transaction network, whose cost for convergence is $O(\frac{1}{\epsilon\alpha})$ controlled by two constant parameters of ϵ and α. Additionally, we give the theoretical proof on how to set the parameters for ensuring our method is able to track the paths among the source node and the target nodes, even in the face of the worst conditions.
- **Evaluation**: We introduce metrics to quantify the effectiveness of transaction tracking methods on Blockchain trading systems, and experimental results demonstrate the priority of the proposed method. Moreover, we collect five standard datasets in Ethereum from real cases for blockchain transaction tracking and anti-money laundering research, and these codes for experiments were published on our Github[1].

The rest of this paper is organized as follows. Section 2 provides the preliminaries about this paper. A general transaction tracking framework Push-Pop model and the Transaction Tracking Rank algorithm are introduced in Section 3. Evaluation and results are presented and discussed in Section 4. In Section 5, we conclude the paper and suggest directions for future work. Citations and appendices are at the end of this paper.

2 PRELIMINARIES

Before introducing our method, this section discusses the problem definition and provides preliminaries for this paper.

2.1 Problem Definition

Transaction records on Blockchain trading systems like Bitcoin and Ethereum represent the relationship of cryptocurrency transfer among accounts. In this paper, transaction records from blockchain trading systems are represented as a transaction network $G = (V, E)$, where V denotes the node set representing addresses, and E denotes the edge set representing transaction relationships between addresses. Considering the amount and timestamp information of the transactions, each edge $e \in E$ can be defined as a four-tuple like (u, v, w, t), where $u, v \in V$ denote the source and target nodes of e, respectively, w denotes the transaction amount, and t denotes the transaction timestamp.

The task of transaction tracking on blockchain can be modeled as graph searching on a transaction network, aiming at finding a subgraph of s named **local tracking network**, denoted by G_s, which contains as many target nodes as possible. As mentioned in the introduction, source nodes are usually deployers of various financial scams such as phishing, Ponzi scheme, blackmail, and extortion scams [16, 24], while the target nodes refer to the addresses which possess the laundered funds awaiting cashing out. Due to the pseudonymous nature of blockchain, it is extremely difficult to pursue the money flows between the source and target nodes and understand real-world entities behind the involved addresses and transactions. In some cases, some addresses are confirmed to be involved in illegal transaction activities by experts, and then

[1] https://github.com/wuzhy1ng/BlockchainSpider

the local tracking network should contain as many of these target nodes as possible. While sometimes, no prior information about target nodes is given, and then the local tracking network aims to contain the addresses with some particular labels (such as exchange deposit addresses) that are related to the source, providing us an opportunity to further infer the target nodes from these labeled nodes.

2.2 Approximately Personalized PageRank

To ensure the effectiveness of the tracking model, we usually need to define an importance measurement of other nodes in the network to the source node. Let $\boldsymbol{p}_s$ denote the personalized PageRank vector for a source node s in a graph $G = (V, E)$, where $\boldsymbol{p}_s(u)$ describes the importance of a node u to the source node s. Let M be the transition matrix, the personalized PageRank vector $\boldsymbol{p}_s$ of a source node s is defined as the unique solution [11] 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, and $M = D^{-1}A$ where D and A are degree matrix and adjacency matrix of G, respectively.

In the original PageRank, $\boldsymbol{p}_s$ is obtained via traditional power iteration [17], which requires an extremely high computational cost in a large network. Therefore, a low-cost method named "local push" procedure [2] was proposed to compute the approximation of personalized PageRank, which starts with all probability residual on the source node of the graph, and pushes the residual to neighbors iteratively in order to get a better estimate of $\boldsymbol{p}_s$. Maintaining residual requires a residual vector $\boldsymbol{r}_s$, where $\boldsymbol{r}_s(u)$ denotes the residual of node u. The residual of node u is transformed into $\boldsymbol{p}_s(u)$ in the local push procedure. 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:

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

where $v \in N(u)$ is the neighbor of u, and the push factor $\theta(u,v)$ between u and v is defined as

$$\theta(u,v) = \frac{1-\alpha}{d(u)}, \tag{3}$$

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

3 PROPOSED APPROACH

This section introduces the Push-Pop model and designs the TTR algorithm in Push-Pop model for blockchain transaction tracking.

3.1 Push-Pop Model

In this paper, we propose a general framework called Push-Pop model for transaction tracking on blockchain trading systems. As shown in Figure 1, the Push-Pop model starts from the source node s with a specific transaction tracking strategy T, builds a witnessed network iteratively, and finds a subgraph from the witnessed network as the local tracking network of the source node finally. Here we aim to find a local tracking network containing as many target nodes as possible. The witnessed network of a source node under a transaction tracking strategy T is defined as $G_s^T = (V_s^T, E_s^T, PR_s^T)$, where V_s^T and E_s^T denote the nodes and edges of G_s^T respectively while PR_s^T is a vector denoting the priority of all nodes in G_s^T.

To build G_s^T for source node s, the steps of Push-Pop model are defined as follows:

(1) *initialize*: Let $V_s^T = \{s\}$, $E_s^T = \emptyset$, and $PR_s^T(s) = 1$.
(2) *pop*: Output the node v with the highest priority from the node set V_s^T.
(3) *expand*: Query the edges $E(v)$ related to v and the neighbor nodes $N(v)$ of v.
(4) *push*: Update the witnessed network G_s^T in which $E_s^T = E_s^T \cup E(v)$, $V_s^T = V_s^T \cup N(v)$, and employ the transaction tracking strategy T to rank the priority of nodes in V_s^T for updating PR_s^T. If the end condition of T is not satisfies, go to step 2.
(5) *extract*: Output a subgraph of the witnessed network G_s^T as the local tracking network G_s.

Obviously, the effectiveness of Push-Pop model depends on the transaction tracking strategy, and how to design an effective strategy will be discussed in the next part.

3.2 Transaction Tracking Rank Algorithm

The TTR algorithm improves the local push procedure of APPR to enable that nodes related to the source node in a given transaction network $G = (V, E)$ can have a higher estimate of $\boldsymbol{p}_s$. According to the attributes of transaction network, we design three local push strategies in the TTR algorithm.

3.2.1 Tracking tendency. Since a transaction relationship between addresses is directed, during the transaction tracking process, the attention to in-degree neighbors and out-degree neighbors may be different. For example, tracking for the target of the fund flows oriented from a particular needs to pay more attention to its out-degree neighbors, while searching for the source of money needs to pay more attention to the in-degree neighbors. Therefore, considering the transaction network G as a directed graph and giving a tracking tendency coefficient $\beta \in [0, 1]$, the out-degree neighbors in a transaction relationship can get a higher estimate of $\boldsymbol{p}_s$ when $\beta > 0.5$, and the in-degree neighbors in a transaction relationship can get a higher estimate of $\boldsymbol{p}_s$ when $\beta < 0.5$. In this way, Equation 1 can be re-written as:

$$\boldsymbol{p}_s = \alpha \boldsymbol{e}_s + (1-\alpha)\boldsymbol{p}_s M_\beta. \tag{4}$$

The transition matrix of Equation 4 is:

$$M_\beta = \beta D_{out}^{-1} A + (1-\beta) D_{in}^{-1} A^T, \tag{5}$$

where A is the adjacency matrix of G, D_{out} is an out-degree matrix of G, and D_{in} is an in-degree matrix of G.

In this way, the local push procedure of node u pushes the residual to itself, out-degree neighbors, and in-degree neighbors. And each in-degree neighbor and each out-degree neighbor receive an equal proportion depending on the in-degree and out-degree, respectively. Therefore, Equations 2 can be re-written as:

$$\begin{cases} \boldsymbol{p}_s(u) = \boldsymbol{p}_s(u) + \alpha \boldsymbol{r}_s(u) \\ \boldsymbol{r}_s(v_{out}) = \boldsymbol{r}_s(v_{out}) + \theta_\beta(u, v_{out})\boldsymbol{r}_s(u) \\ \boldsymbol{r}_s(v_{in}) = \boldsymbol{r}_s(v_{in}) + \theta_\beta(u, v_{in})\boldsymbol{r}_s(u) \end{cases}, \tag{6}$$

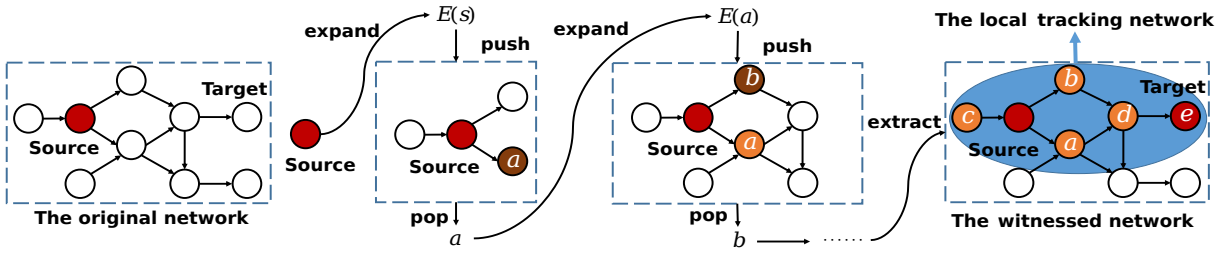

Figure 1: Push-Pop Model, a general transaction tracking framework.

where $v_{out} \in N_G^{out}(u)$ and $v_{in} \in N_G^{in}(u)$ denote out-degree neighbor and in-degree neighbor of u respectively, and the push factors are defined as follows:

$$\theta_\beta(u, v_{out}) = \frac{(1-\alpha)\beta}{d_{out}(u)}, \tag{7}$$

$$\theta_\beta(u, v_{in}) = \frac{(1-\alpha)(1-\beta)}{d_{in}(u)}, \tag{8}$$

in which $d_{out}(u)$ and $d_{in}(u)$ denote out-degree and in-degree of u respectively. And if a node u has no out-degree neighbors or in-degree neighbors, the residual can be pushed to itself.

3.2.2 Weight pollution. Traditional APPR pushes the residual of a node to the neighbors by considering an equal weight during a local push iteration. However, in a transaction relationship, the transaction strength is usually weighted by the transaction amount. For each node in the network, a neighbor who has transaction relationships with a larger amount of money is considered to be more relevant to the node. Therefore, by considering the transaction network G as a weighted directed graph with the transaction amount information as the weight, Equation 4 can be re-written as:

$$\boldsymbol{p}_s = \alpha \boldsymbol{e}_s + (1-\alpha)\boldsymbol{p}_s M_w. \tag{9}$$

The transition matrix of Equation 9 is:

$$M_w = \beta \tilde{D}_{out}^{-1} W + (1-\beta)\tilde{D}_{in}^{-1} W^T, \tag{10}$$

where W is the weighted adjacency matrix of G, $\tilde{D}_{out}$ is the weighted out-degree matrix of G, $\tilde{D}_{in}$ is the weighted in-degree matrix of G. The diagonal elements of $\tilde{D}_{out}$ and $\tilde{D}_{in}$ are defined as follows:

$$\tilde{D}_{out}(i,i) = ||W(i,)||, \tag{11}$$

$$\tilde{D}_{in}(i,i) = ||W^T(i,)||, i \le |V| \tag{12}$$

where $W(i,)$ and $W^T(i,)$ denote the i-th row in W and W^T respectively. All off-diagonal elements in $\tilde{D}_{out}$ and $\tilde{D}_{in}$ equal to 0.

In this way, the local push procedure of node u pushes residual to itself, the out-degree neighbors and the in-degree neighbors with different ratios determined by edge weight. And Equations 6 can be re-written as:

$$\begin{cases} \boldsymbol{p}_s(u) = \boldsymbol{p}_s(u) + \alpha \boldsymbol{r}_s(u) \\ \boldsymbol{r}_s(v_{out}) = \boldsymbol{r}_s(v_{out}) + \theta_w(u, v_{out})\boldsymbol{r}_s(u) \\ \boldsymbol{r}_s(v_{in}) = \boldsymbol{r}_s(v_{in}) + \theta_w(u, v_{in})\boldsymbol{r}_s(u) \end{cases}, \tag{13}$$

where the push factors are defined as follows:

$$\theta_w(u, v_{out}) = (1-\alpha)\beta \frac{w(u, v_{out})}{\tilde{d}_{out}(u)}, \tag{14}$$

$$\theta_w(u, v_{in}) = (1-\alpha)(1-\beta)\frac{w(v_{in}, u)}{\tilde{d}_{in}(u)}, \tag{15}$$

in which $w(u, v)$ for $\forall u, v \in V$ denotes the transaction amount from u to v, and $\tilde{d}_{out}(u)$ and $\tilde{d}_{in}(u)$ denote the weighted out-degree and weighted in-degree of u respectively.

3.2.3 Temporal reasoning. Blockchain transactions are recorded in blocks chronologically, and each block contains a specific timestamp. Since the fund flow transferring process is dynamic, the transaction time information can be taken into consideration in the local push procedure. In the scenario of blockchain transaction tracking, tracking for the target of a fund flow usually follows the paths with increasing timestamps, while tracking back to the source of a fund flow follows the paths with decreasing timestamps. Therefore, considering the transaction network G as a temporal weighted directed graph in which $G^{(t)}$ denotes the weighted directed graph at timestamp t, for all timestamp $\{t_1, ..., t_k, ..., t_n\}$ in G, Equation 9 can be re-written as:

$$\boldsymbol{p}_s^{(t)} = \alpha \boldsymbol{e}_s^{(t)} + (1-\alpha)p_s^{(t)} M_w^{(t)}, \tag{16}$$

where $\boldsymbol{p}_s^{(t)}$ denotes the estimate of $\boldsymbol{p}_s$ at timestamp t, $M_w^{(t)}$ denotes the transition matrix at timestamp t, and $\boldsymbol{e}_s^{(t)}$ denotes the indicator vector at timestamp t in which $\boldsymbol{e}_s^{(t_1)} = \boldsymbol{e}_s$ and $\boldsymbol{e}_s^{(t_k)} = \boldsymbol{p}_s^{(t_{k-1})}$. In this case, $\boldsymbol{p}_s$ is redefined as follow:

$$\boldsymbol{p}_s = \frac{1}{n}\sum_{t'=t_1}^{t_n} \boldsymbol{p}_s^{(t')}. \tag{17}$$

The transition matrix of Equation 16 is:

$$M_w^{(t)} = \beta \hat{D}_{out}^{(t)^{-1}} W^{(t)} + (1-\beta)\hat{D}_{in}^{(t)^{-1}} W^{(t)^T}, \tag{18}$$

where $W^{(t)}$ is a weighted adjacency matrix at timestamp t, $\hat{D}_{out}^{(t)}$ and $\hat{D}_{in}^{(t)}$ denote the cumulative weighted out-degree matrix and the cumulative weighted in-degree matrix at timestamp t respectively. The nonzero elements of $\hat{D}_{out}$ and $\hat{D}_{in}$ are defined as follows:

$$\hat{D}_{out}^{(t)}(i,i) = \sum_{t' \le t} ||W^{(t')}(i,)||, \tag{19}$$

$$\hat{D}_{in}^{(t)}(i,i) = \sum_{t' \ge t} ||W^{(t')}(,i)||, t_1 \le t \le t_n, i \le |V|, \tag{20}$$

in which $W^{(t')}(i,)$ and $W^{(t')}(,i)$ denote the i-th row and the i-th column in the weighted adjacency matrix at timestamp t' respectively.

Algorithm 1 Transaction Tracking Rank

Input: Source node s, transactions network $G = (V, E)$ with timestamp range $\{t_1, t_2, ..., t_n\}$, teleport constant α, minimum error ϵ, and tracking tendency coefficient β.
Output: the estimate of $\boldsymbol{p}_s$

$\boldsymbol{p}_s(s) = \alpha$
$R_s(t, v) = (1-\alpha)\beta w / \tilde{d}_{out}(s)$ for $\forall(s, v, w, t) \in E_{out}(s)$
$R_s(t, v) = (1-\alpha)(1-\beta) w / \tilde{d}_{in}(s)$ for $\forall(v, s, w, t) \in E_{in}(s)$
if $E_{out}(s) = \emptyset$ **then**
  $R_s(t_1, s) = (1-\alpha)\beta$
end if
if $E_{in}(s) = \emptyset$ **then**
  $R_s(t_n, s) = (1-\alpha)(1-\beta)$
end if
while $||R_s(u)|| \geq \epsilon, u \in V$ **do**
  Let $R'_s = R_s$, and $R_s(u) = \vec{0}$.
  Apply R'_s to update $\boldsymbol{p}_s$ and R_s through selfPush, forwardPush and backwardPush.
end while
return $\boldsymbol{p}_s$

In this way, the local push procedure of node u pushes the residual to itself, out-degree neighbors, and in-degree neighbors with different ratios determined by the transaction amount and time. Equations 13 can be re-written as:

$$\begin{cases} \boldsymbol{p}_s(u) = \boldsymbol{p}_s(u) + \alpha \boldsymbol{e} \cdot R_s(u) \\ R_s(v_{out}) = R_s(v_{out}) + \theta_t(u, v_{out}) \odot R_s(u) \\ R_s(v_{in}) = R_s(v_{in}) + \theta_t(u, v_{in}) \odot R_s(u) \end{cases}, \tag{21}$$

where R_s denotes a $n \times |V|$ matrix in which $R_s(v_i) = R_s(, i)$ denotes the residual of a node v_i at each timestamp, $\odot$ denotes the element-wise product, and the push factors are defined as follows:

$$\theta_t(u, v_{out}) = (1-\alpha)\beta w_t(u, v_{out}) \oslash \hat{d}_{out}(u), \tag{22}$$

$$\theta_t(u, v_{in}) = (1-\alpha)(1-\beta) w_t(v_{in}, u) \oslash \hat{d}_{in}(u), \tag{23}$$

in which $w_t(u, v)$ for $\forall u, v \in V$ is a vector denoting the transaction amount from u to v at each timestamp, $\hat{d}_{out}(u)$ and $\hat{d}_{in}(u)$ denote the temporal weight out-degree and in-degree at each timestamp respectively, and $\oslash$ denotes the element-wise division.

Based on the above local push strategies, we propose the TTR algorithm. Giving a transactions network $G = (V, E)$, TTR tracking for the target starting from the source node s. Initializing R_s with edges related to s, TTR calls the local push procedure to push the residual for a node itself, the out-degree neighbors, and the out-degree neighbors by "selfPush", "forwardPush" and "backwardPush" procedure respectively, until the residual of each node is within the minimum error ϵ. Algorithm 1 describes the framework of the TTR algorithm, Algorithm 2, Algorithm 3 and Algorithm 4 describe the "selfPush", "forwardPush" and "backwardPush" procedure in detail respectively.

Here are some useful theorems (also see [2]) to describe the cost of TTR algorithm, and the proofs given in appendices.

Theorem 3.1. *Line 10-line 13 in Algorithm 1, runs in $O(\frac{1}{\epsilon\alpha})$ time, and the number of nodes with non-zero values in the output TTR vector is at most $O(\frac{1}{\epsilon\alpha})$.*

Algorithm 2 selfPush

Input: Node u, temporal residual R'_s, estimate $\boldsymbol{p}_s$, and teleport constant α.

for $t \in [t_1, t_n]$ **do**
  $\boldsymbol{p}_s(u) += \alpha R'_s(t, u)$
end for

Algorithm 3 forwardPush

Input: Node u, out-edges $E_{out}(u)$, temporal residual R'_s and R_s, teleport constant α, and tracking tendency coefficient β.

for $\forall e : (u, v, w, t) \in E_{out}(u)$ **do**
  $\Delta = 0$
  for $\tau \in [t_1, t]$ **do**
    $\Delta += w R'_s(\tau, u) / \hat{d}^{(\tau)}_{out}(u)$
  end for
  $R_s(t, v) += (1-\alpha)\beta\Delta$
end for
Calculate $max(t)$ from all t in $E_{out}(u)$.
for $\tau \in [max_t, t_n]$ **do**
  $R_s(\tau, u) += (1-\alpha)\beta R'_s(\tau, u)$
end for

Algorithm 4 backwardPush

Input: Node u, in-edges $E_{in}(u)$, temporal residual R'_s and R_s, teleport constant α, and tracking tendency coefficient β.

for $\forall e : (v, u, w, t) \in E_{in}(u)$ **do**
  $\Delta = 0$
  for $\tau \in [t, t_n]$ **do**
    $\Delta += w R'_s(\tau, u) / \hat{d}^{(\tau)}_{in}(u)$
  end for
  $R_s(t, v) += (1-\alpha)(1-\beta)\Delta$
end for
Calculate $min(t)$ from all t in $E_{in}(u)$.
for $\tau \in [t_1, min(t)]$ **do**
  $R_s(\tau, u) += (1-\alpha)(1-\beta)R'_s(\tau, u)$
end for

Theorem 3.2. *The cost of local push procedure including selfPush, forwardPush and backwardPush is $O(|E(u)|log(|E(u)|)$.*

3.3 TTR-based Push-Pop Model

The TTR-based Push-Pop model uses the TTR algorithm as the transaction tracking strategy, which can track the fund flow of the source node s in a transaction network, calculate the relevance between other nodes in the network and the source node, and output a local tracking network.

For initialization, the TTR-based Push-Pop model set $V_s^T = \{s\}$, and then calls *pop*, *expand*, and *push* in turns until $||R_s(u)|| < \epsilon$ for $\forall u \in V_s^T$. Finally, the TTR-based Push-Pop model calls *extract* for outputting a local tracking network. The procedures of the TTR strategy in TTR-based Push-Pop model are as follows:

- *expand*: Query the related edges for a given node u.

Algorithm 5 TTR-based Local Community Discovery

Input: The source node s, the witnessed network $G_s^T = (V_s^T, E_s^T)$
Output: the local tracking network

1: $S = \{s\}$
2: $\boldsymbol{p}_s = TransactionTrackingRank(s, G_s^T, \epsilon, \alpha, \beta)$
3: **while** $\Phi(S) \geq \varphi$ **do**
4: $\quad u = maxEstimateNode(\boldsymbol{p}_s)$
5: $\quad S = S \cup \{u\}$
6: **end while**
7: **return** $G_s^T.subgraph(S)$

- *push*: The TTR strategy receives a series of edges related to a node u for updating the witnessed network G_s^T, and calls the local push procedure for u.
- *pop*: Output a node that has a maximum residual from V_s^T.
- *extract*: Output a local tracking network through the TTR-based Local Community Discovery Algorithm 5.

Aiming at extracting the most important partition of G_s^T, the TTR-based Local Community Discovery Algorithm 5 constructs a local community as the local tracking network. Here, we define a subgraph G_{sub} of G_s^T with a low conductance as the local community according to previous work [2, 3], in which its conductance is:

$$\Phi(S) = \frac{\boldsymbol{p}_s(\partial(S))}{\boldsymbol{p}_s(S)} \tag{24}$$

where $S \subseteq V_s^T$ denotes the nodes of G_{sub}, the boundary $\partial(S) = \{v|(u,v) \in E_t \wedge u \in S \wedge v \in \bar{S}\}$, $\bar{S}$ is the complement of S and $\boldsymbol{p}_s(S) = \sum_{u \in S} \boldsymbol{p}_s(u)$ denotes the sum of $\boldsymbol{p}_s$ over each node in S. In this way, a subgraph G_{sub} of G_s^T is a local community satisfying:

$$\Phi(S) < \varphi, \tag{25}$$

where φ denotes the maximum conductance of S. For initialization, algorithm 5 set $S = \{s\}$, and then adds a node with the maximum estimate of $\boldsymbol{p}_s$ in V_s^T on each step until satisfying Equation 25.

Here are two properties of the TTR-based Push-Pop model and their proofs are given in appendices.

Proposition 3.1. *A n-order neighbor [12] of the source node can be found in V_s^T, in which n satisfies:*

$$n \leq \frac{log(\epsilon)}{log(1-\alpha)}. \tag{26}$$

This proposition estimates what depth can the TTR-based Push-Pop model track in a network from the source node.

Proposition 3.2. *At least one path between the source node s and a n-order target node v_n can be found in local tracking network of TTR-based Push-Pop model under the worst conditions, which requires:*

$$\begin{cases} \varphi \geq \epsilon \\ \bar{d}\varphi^{\frac{1}{n}} \leq (1-\alpha) \cdot min(\beta, 1-\beta) \end{cases}, \tag{27}$$

where $\bar{d}$ denotes the average degree of nodes in the paths among the source node and target nodes.

Let n denotes the nodes number of a local transaction network, we suggest that $\epsilon = \Omega(\frac{1}{n})$ [29]. The proposition 3.2 limits the value range of each parameter in Algorithm 5.

Table 1: Statistics of Cases

Case	Source[1]	#Target	#Node	#Edge	Block[2]
PlusToken	0xf4a2e	17	68	164	7993213
TokenStore	0x068ac	5	80	194	7802134
Cryptopia	0xd4e79	4	9	24	9128188
Kucoin	0xeb319	-	-	-	10933499
Upbit	0xa0987	-	-	-	9007863

[1] Here provides the address prefix of the source node.
[2] Here provides the block number of the first transaction related to the cases.

4 EXPERIMENTS

In this section, we compare several blockchain transaction tracking baselines with real-world transaction tracking cases to verify the effectiveness of our method. Besides, we conduct case study with network visualization techniques.

4.1 Experimental Setup

4.1.1 Experimental Cases. We collect five transaction tracking cases in Ethereum for our experiments, which are published by CoinHolmes [19] from an industry leading blockchain security company named Peckshield. The statistics of each case are shown in Table 1, which includes the source, number of targets, number of nodes, number of edges, and the block number of the first transaction related to the case. And the transaction tracking results of the cases given the number of targets have been announced by experts. Among these cases, **Plustoken** and **TokenStore** are proved to be Rug Pull projects, and their tracking targets have been given by experts. More than 300 thousand illegal ETH are related to these two Rug Pull projects. The **Cryptopia** exchange was attacked by hackers in May 2019, and the related target nodes have also been given by experts. More than 31.8K ETH was stolen from this attack according to CoinHolmes. The **Kucoin** exchange and the **Upbit** exchange were attacked by hackers in September 2020 and November 2019 respectively with a large amount of cryptocurrency flowing out. Experiments are started from the source node in these cases to find the local tracking network.

4.1.2 Compared Methods. We compare our method with several baseline blockchain transaction tracking methods, including:

- **BFS**: Breadth-First Search, which is the first and the most commonly used transaction tracking method.
- **Poison** [13]: A kind of taint analysis technology in blockchain transaction tracking. 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 tracking. Each output of a transaction with a dirty input is considered to be tainted partially according to the amount value in this method.
- **APPR** [2]: The Approximately Personalized PageRank algorithms, which can calculate the importance of a node in a network to the given source node with extremely low cost.
- **TTR**: The method we proposed with three local push strategies, and we abbreviate the TTR algorithm with tracking tendency, weight pollution, and temporal reasoning strategies as TTR-base, TTR-weight, and TTR-time respectively.

Table 2: Experimental Results.

	PlusToken			TokenStore			Cryptopia			Kucoin			Upbit		
Methods	D_l	K	R	D_l	K	R	D_l	K	R	D_l	K	P_l	D_l	K	P_l
BFS	4.0×10^{-9}	3	**0.93**	1.6×10^{-7}	3	0.4	3.4×10^{-7}	3	1.0	1.1×10^{-8}	3	-	3.8×10^{-9}	3	-
Poison	6.3×10^{-9}	3	0	1.6×10^{-7}	3	0.4	2.3×10^{-6}	3	1.0	2.3×10^{-8}	3	-	6.4×10^{-6}	3	-
Haircut	7.6×10^{-9}	**6**	**0.93**	8.1×10^{-8}	**6**	**1.0**	7.0×10^{-8}	7	**1.0***	2.0×10^{-5}	4	-	4.0×10^{-7}	5	-
APPR	0	0	0	8.1×10^{-4}	5	0.4	3.1×10^{-4}	7	**1.0***	6.0×10^{-4}	4	0.25	2.4×10^{-4}	4	0.26
TTR-base	4.9×10^{-5}	5	**0.93**	$\mathbf{2.0\times10^{-3}}$	5	0.4	4.0×10^{-4}	6	**1.0***	$\mathbf{6.2\times10^{-4}}$	4	0.26	2.8×10^{-4}	6	0.44
TTR-weight	2.2×10^{-5}	**6**	**0.93**	2.8×10^{-4}	**6**	**1.0**	6.9×10^{-4}	7	**1.0***	2.7×10^{-4}	5	0.39	4.0×10^{-4}	**8**	**0.75**
TTR-time	$\mathbf{1.2\times10^{-4}}$	5	**0.93**	1.6×10^{-4}	**6**	**1.0**	$\mathbf{9.6\times10^{-4}}$	**10**	**1.0***	2.2×10^{-4}	**7**	**0.52**	$\mathbf{4.3\times10^{-4}}$	**8**	0.65

* These methods can find some extra suspicious target nodes, referring to Section 4.3.1.

The data for experiments come from Etherscan[2], and we limit the start query block as the block containing the first transactions related to a specific case.

4.1.3 Experimental Settings. Based on the experiences, BFS and Poison track the 3-order neighborhood of the source node since the increase of order leads to the exponential growth of the local tracking network size in these two methods, bringing great difficulty to transaction auditing.

In addition, we use Haircut to track the "dirty money" from the source node until the amount proportion of "dirty money" of all nodes in the local tracking network is less than 0.1% of that from the source node. Moreover, we set $\alpha = 0.15$, $\epsilon = 10^{-4}$ for APPR and TTR, and $\varphi = 10^{-4}$, $\beta = 0.7$ for TTR to ensure that our method is able to find the path among the source node and the target nodes in the 4-order neighborhood at least according to the Proposition 3.2, since the average node degree of the local tracking networks in these cases is under 2.5.

4.1.4 Metrics. We use the following metrics to measure the extracted local tracking networks and compare the performance of different methods, including:

- **Label density** D_l: A local tracking network with more labeled nodes and smaller network size is easier for expert verification. Therefore, we define the label density metric as:

$$D_l = \frac{|V_l|}{|V|^2}, \tag{28}$$

  where $|V_l|$ is the number of labeled nodes and $|V|$ denotes the number of nodes in a local tracking network.
- **Tracking depth** K: The metric measures how deep can a transaction tracking algorithm traverse the network from a source node, indicating that up to K-order neighbors of the source node are included in the local tracking network.
- **Recall** R: The recall metric evaluates how many labeled targets can be detected by an algorithm, which is defined as:

$$R = \frac{|V_t|}{|\bar{V}_t|}, \tag{29}$$

  where $|V_t|$ is the number of target nodes in the local tracking network output by a transaction tracking method and $|\bar{V}_t|$ is the number of target nodes in a case.
- **Relevance estimate of labels** P_l: In cases without verified target nodes given before, the effectiveness of an APPR-based transaction tracking method can be verified by the estimate $\boldsymbol{p}_s(u)$ of each labeled node u, which is defined as:

$$P_l = \sum_{u \in V_l} \boldsymbol{p}_s(u), \tag{30}$$

  where V_l denotes a labeled node set of the local tracking network. And a higher P_l indicates that the tracking clues are clearer in a local tracking network.

4.2 Experimental Result

Table 2 shows the experimental results of the five cases, from which we can obtain the following observations.

Firstly, BFS and Poison output the local tracking networks with a low recall and a low label density for all cases, which bring a great difficulty to transaction auditing. ***Secondly***, the local tracking networks of Haircut get a higher recall and a greater tracking depth in all cases, but the low label density of the network also indicates the huge size of the output local tracking network, which is hard for transaction auditing. ***Thirdly***, the recall of APPR is lower than the above methods. However, it is worth mentioning that the APPR can get a higher label density in most cases. Especially, since the residual of a node is pushed to its neighbors equally in the local push iteration, APPR converges quickly and results in smaller local tracking networks in these cases. And in the case of PlusToken, the local tracking network of APPR contains the source node merely. **Fourthly**, our proposed TTR with different local push strategies can obtain a better performance than APPR, in which TTR-weight obtains a higher tracking depth and a higher relevance estimate of labels in PlusToken, Kucoin, and Upbit. In addition, TTR-weight obtains further improvement over the tracking depth and the relevance estimate of labels since it takes the transaction weight into consideration. Furthermore, using time reasoning, TTR-time gets the best performance in most cases.

Since the estimate of $\boldsymbol{p}_s$ on a node represents the relevance relationship between this node and the source, we can audit the nodes in the local tracking network according to the descending order of $\boldsymbol{p}_s$. To compare the APPR-based methods, we take out the top n most important nodes to the source node with APPR, TTR-base, TTR-weight, and TTR-time, and calculate the average recall in the case of PlusToken, TokenStore, and Cryptopia for different

[2]https://etherscan.io

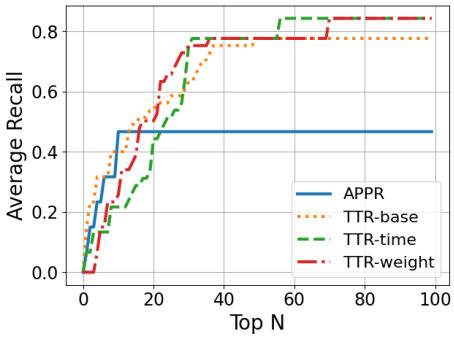

Figure 2: Top n important nodes and average recall

n. The result is displayed in Figure 2, where the TTR methods can achieve 31% over gain over the recall of APPR when $n > 50$. Compared with TTR-base, TTR-weight and TTR-time can obtain a larger recall when $n > 70$.

4.3 Case Study

In this part, we visualize the money tracking flows on a case with verified labels and a case without verified labels with the TTR-weight algorithm with Gephi 0.9.2 [5], in order to evaluate the feasibility of the proposed method.

4.3.1 Cryptopia. The Cryptopia exchange was attacked by hackers in May 2019. According to the local tracking network published by CoinHolmes [3], the source node with prefix 0xd4e79 stole 31.8K ETH from Cryptopia and then transferred about 10K ETH to 4 target nodes labeled with EtherDelta.

Figure 3 presents the transaction tracking result with the TTR-weight method, where the source node and the target nodes are marked with labels. Additionally, the node size is proportional to its estimate of $\boldsymbol{p}_s(u)$ for each node u, so that the higher $\boldsymbol{p}_s(u)$ is, the larger the node diameter is. Based on the local tracking network of our method, we can find 4 target nodes labeled as EtherDelta (an exchange) in the 2-order neighborhood of the source node easily, which is consistent with the results in CoinHolmes.

Moreover, another 4-order neighbor of the source node, labeled as Yobit.net with the address prefix 0xf5bec, can also be found in the figure, which is not discovered by CoinHolmes. In fact, Yobit.net is an exchange. According to the transaction pattern and the fund flow path shown in the figure, it's reasonable that the hackers cashed out a part of the stolen ETH via Yobit.net, so that this node is likely to be a target node. And from these fund flow paths, we can find the other 1420 stolen ETH.

Besides, we can find two nodes labeled as OKEx in the figure, which is the deposit address of the OKEx exchange. According to the tracking ETH flows in the figure, it's reasonable that the hackers cashed out another part of the stolen ETH via OKEx, which is about 18.47K ETH. Therefore, more than 94% of the stolen ETH of Cryptopia can be found with our proposed TTR-weight algorithm in this case.

[3]https://trace.coinholmes.com

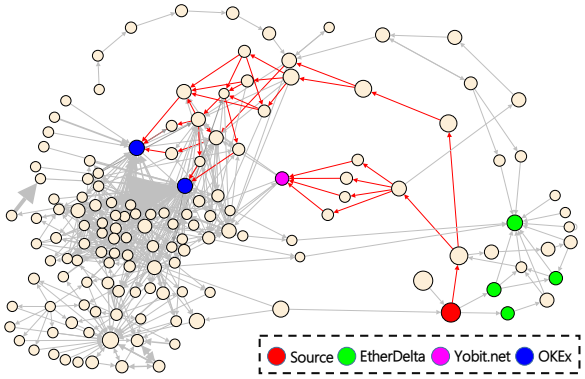

Figure 3: Visualization of the TTR result for Cryptopia

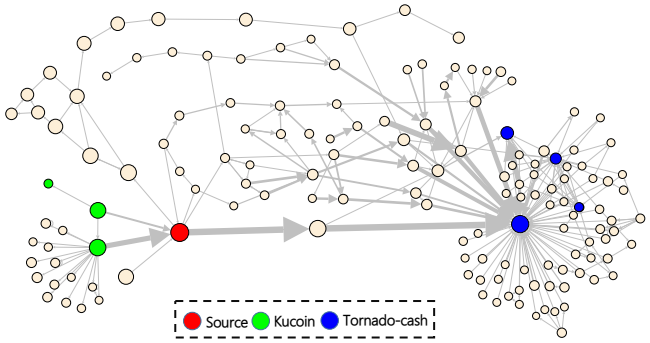

Figure 4: Visualization of the TTR result for Kucoin

4.3.2 Kucoin. The Kucoin exchange was attacked by hackers in September 2020, and there is still a large amount of stolen funds that have not been tracked until now. As shown in Figure 4, the source node of 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 [4].

Through the local tracking network 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 Peckshield. 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 track the downstream fund flow when money is transferred into Tornado Cash. Therefore, some researchers have conducted analysis on Tornado Cash in recent years [6].

5 CONCLUSION

In this paper, we discussed the challenges and problems of transaction tracking on blockchain trading systems and proposed a general transaction tracking framework named as Push-Pop model. Specifically, we designed the Transaction Tracking Rank (TTR) algorithm for Push-Pop model to realize effective and low-cost transaction tracking. Finally, a series of theoretical and experimental results on five realistic cases on Ethereum demonstrated the priority of the proposed method over existing methods.

In our future work, we will further delve into transaction tracking on blockchain by integrating more transaction features like bytecode and logs and design effective methods for some other blockchain systems with a privacy-enhancing mechanism.

ACKNOWLEDGMENTS

REFERENCES

[1] Serge Abiteboul, Mihai Preda, and Gregory Cobena. 2003. Adaptive on-line page importance computation. In *Proceedings of the 12th International Conference on World Wide Web*. ACM, Budapest, Hungary, 280–290.

[2] 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.

[3] 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.

[4] ayefda. 2021. Introduction of Tornado.Cash. https://docs.tornado.cash/.

[5] 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.

[6] 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).

[7] Binance. 2021. Binance chain documentation. https://docs.binance.org/.

[8] Chainalysis. 2017. The rise of cybercrime on Ethereum. https://blog.chainalysis.com/reports/the-rise-of-cybercrime-on-ethereum.

[9] Chainalysis. 2020. The Chainalysis crypto crime report is here. Download to learn why 2019 was the year of the Ponzi scheme. https://blog.chainalysis.com/reports/cryptocurrency-crime-2020-report.

[10] 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.

[11] Taher H Haveliwala. 2003. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. *IEEE transactions on knowledge and data engineering* (2003), 784–796.

[12] Dan Lin, Jiajing Wu, Qi Yuan, and Zibin Zheng. 2020. Modeling and understanding Ethereum transaction records via a complex network approach. *IEEE Transactions on Circuits and Systems II: Express Briefs* (Dec. 2020), 2737–2741.

[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] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf.

[15] Shen Noether and Sarang Noether. 2014. *Monero is not that mysterious*. Technical Report. Monero Research Lab.

[16] Frédérique Oggier, Anwitaman Datta, and Silivanxay Phetsouvanh. 2020. An ego network analysis of sextortionists. *Social Network Analysis and Mining* (June 2020), 1–14.

[17] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web*. Technical Report. Stanford InfoLab.

[18] Peckshield. 2021. The 2021 digital currency AML research report. https://coinholmes.com/static/pdf/2021_1.pdf.

[19] Peckshield. 2021. CoinHolmes. https://trace.coinholmes.com/.

[20] 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.

[21] 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).

[22] Gavin Wood et al. 2014. *Ethereum: A secure decentralised generalised transaction ledger*. Technical Report. ETHCore.

[23] 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.

[24] Jiajing Wu, Jieli Liu, Yijing Zhao, and Zibin Zheng. 2021. Analysis of cryptocurrency transactions from a network perspective: An overview. *Network and Computer Applications* (2021), 103139.

[25] 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.

[26] Jennifer J Xu and Hsinchun Chen. 2004. Fighting organized crimes: Using shortest-path algorithms to identify associations in criminal networks. *Decision Support Systems* (2004), 473–487.

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

[28] 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.

[29] Hongyang Zhang, Peter Lofgren, and Ashish Goel. 2016. Approximate personalized pagerank on dynamic graphs. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, San Francisco, CA, USA, 1315–1324.

[30] 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.

[31] Lin Zhao, Sourav Sen Gupta, Arijit Khan, and Robby Luo. 2021. Temporal analysis of the entire Ethereum blockchain network. In *Proceedings of the Web Conference*. Association for Computing Machinery, New York, NY, USA, 2258–2269.

A APPENDICES

A.1 Proof of Theorem 3.1

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

A.2 Proof of Theorem 3.2

Proof. The residual of a node u coming from its neighbors ensures that the number of non-zero elements in $R_s(u)$ are less than $|E(u)|$, where $E(u)$ contains edges related to u, so the cost of selfPush will not exceed $O(E(u))$. In addition, forwardPush and backwardPush are both related to the timestamp of edges, so all edges among node u and its neighbors can be sorted according to their timestamp firstly, whose cost is $O(|E(u)|log(|E(u)|))$, and then the summation terms are calculated in a timestamp order, whose cost is $O(|E(u)|)$. Therefore, the cost of the local push procedure is:

$$O(|E(u)|log(|E(u)|) + |E(u)|) = O(|E(u)|log(|E(u)|)). \tag{31}$$

□

A.3 Proof of Proposition 3.1

Proof. In order to maximize the estimate of $\boldsymbol{p}_s$ for the nodes having a high order of 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. Considering $\beta = 1$ here, the sum of residual pushed from the source node s to the neighbors of order 1 to order n are:

$$\begin{cases} r^{(1)} = (1-\alpha), \\ r^{(2)} \geq (1-\alpha)(r^{(1)} - k_1\epsilon) = (1-\alpha)^2 - (1-\alpha)k_1\epsilon, \\ \cdots\cdots \\ r^{(n)} \geq (1-\alpha)(r^{(n-1)} - k_{n-1}\epsilon) \\ \qquad = (1-\alpha)^n - \sum_{i=1}^{n-1}(1-\alpha)^{n-i}k_{n-i}\epsilon, \end{cases} \tag{32}$$

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:

$$\sum_{i=1}^{n-1}(1-\alpha)^{n-i}k_{n-i}\epsilon = 0, \tag{33}$$

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

$$r^{(n)} = (1-\alpha)^n \leq \epsilon \Rightarrow n \leq \frac{log(\epsilon)}{log(1-\alpha)}. \tag{34}$$

□

A.4 Proof of Proposition 3.2

Proof. Consider the end condition of TTR-based Local Community Discovery Algorithm:

$$\Phi(S) = \frac{\boldsymbol{p}_s(\partial(S))}{\boldsymbol{p}_s(S)} < \varphi, \tag{35}$$

and the selfPush procedure at the first iteration of TTR Algorithm ensures:

$$\boldsymbol{p}_s(s) \geq \alpha. \tag{36}$$

Since $s \in S$, then $\boldsymbol{p}_s(S) \geq \alpha$, so:

$$\boldsymbol{p}_s(\partial(S)) < \varphi\alpha, \tag{37}$$

which means that the path between s and v_n must be included in the local tracking network when $\boldsymbol{p}_s(v_n) \geq \varphi\alpha$. In this way, $\boldsymbol{p}_s(v_n)$ can obtain a minimum value with only one selfPush operation for v_n, and then $\boldsymbol{p}_s(v_n) \geq \varphi\alpha$ needs:

$$\alpha||R_s(v_n)|| \geq \varphi\alpha \Rightarrow ||R_s(v_n)|| \geq \varphi \tag{38}$$

Moreover, in order to ensure that at least one selfPush operation on v_n, the residual of v_n must be greater than ϵ, that is:

$$\varphi \geq \epsilon. \tag{39}$$

Consider that satisfying Equation 38 in the worst case: 1) The path between the source node and a target node is weakly connected, which makes the tracking tendency invalid. 2) The edges related to each node on the paths among the source node and the target node have the same weight, which makes the weight pollution invalid. 3) Each path between the source node and a target node without increasing or decreasing timestamp, which makes time reasoning invalid. 4) There is only one path between the source node and a target node. In this case, in order to satisfy the Equation 38, the residual sum of m-order neighbors ($m \leq n$) of the target node v_n are:

$$\begin{cases} r^{(n)} \geq \varphi, \\ r^{(n-1)} \geq \frac{\varphi d_{n-1}}{min(\beta,1-\beta)(1-\alpha)}, \\ r^{(n-2)} \geq \frac{\varphi d_{n-1} d_{n-2}}{min^2(\beta,1-\beta)(1-\alpha)^2}, \\ \cdots\cdots \\ r^{(n-m)} \geq \frac{\varphi \prod_{i=1}^{m} d_{n-i}}{min^m(\beta,1-\beta)(1-\alpha)^m}, \end{cases} \tag{40}$$

where d_{n-i} denotes the degree of the $n-i$ order node in this path. Consider the inequality as follow:

$$\prod_{i=1}^{m} d_{n-i} \leq (\frac{1}{m}\sum_{i=1}^{m} d_{n-i})^m \to \bar{d}^m. \tag{41}$$

If the m-order neighbor of the target node is the source node, then:

$$1 \geq \frac{\varphi \bar{d}^n}{min^n(\beta, 1-\beta)(1-\alpha)^n} \Rightarrow \bar{d}\varphi^{\frac{1}{n}} \leq (1-\alpha)\cdot min(\beta, 1-\beta) \tag{42}$$

□