Analysis of Address Linkability in Tornado Cash on Ethereum

Yujia Tang¹, Chang Xu^{2(✉)}, Can Zhang², Yan Wu², and Liehuang Zhu²

¹ School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China

² School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China
xuchang@bit.edu.cn

Abstract. Tornado Cash, the most popular non-custodial coin mixer on Ethereum, is widely used to protect the privacy of addresses. However, some inappropriate transaction behaviors in Tornado Cash mixing mechanism lead to the risk of privacy leakage. More specifically, the malicious attackers can link multiple addresses of the same users according to the transaction data. Motivated by the above problem, this paper systematically analyzes the privacy issues of Tornado Cash for the first time. In this paper, we give the macroscopic analysis of Tornado Cash based on the on-chain data and formalize two types of transaction patterns. Focus on the presented transaction patterns, we propose three heuristic clustering rules to link the users' addresses, which reduce the size of users' anonymity set. Finally, we perform the experiment on real Tornado Cash transaction data to describe the effectiveness of the proposed clustering rules.

Keywords: Ethereum · Tornado Cash · Address linkability · Heuristic cluster

1 Introduction

Ethereum is the second most valuable cryptocurrency which has been widely used in various areas. In Ethereum blockchain, the transaction data is publicly stored on all Ethereum full nodes, which means any node can access the complete information from the chain. Unfortunately, malicious users can also analyze it to infer correlations between addresses and even the identity of other users with the help of background knowledge.

With the increasing attention to privacy protection, more and more users are trying to hide the association between their own addresses using privacy preservation mechanisms such as coin mixing [3, 8–12, 15].

Supported by the National Key Research and Development Program of China (Grant No. 2020YFB1006101), the Key-Area Research and Development Program of Guangdong Province (No. 2019B010137003) and the National Natural Science Foundation of China (Grant No. 61972037).

© The Author(s) 2022

W. Lu et al. (Eds.): CNCERT 2021, CCIS 1506, pp. 39–50, 2022.

https://doi.org/10.1007/978-981-16-9229-1_3

Tornado Cash is one of the most popular coin mixing tools on Ethereum, which had nearly 1.6 million Ether flowing into Tornado coin mixing contracts worth over 2.4 billion in USD. And its handling fee has reached a high of \$2.82 million¹. Users on Tornado Cash only need to invoke the relevant smart contract to complete the coin mixing process. It provides huge convenience for users to enhance the privacy of address linkability. However, some inappropriate use has made it much less effective and exposed some security concerns.

This paper presents the first heuristic address correlation clustering approach based on the user’s behaviour in the Tornado Cash coin mixing scenario, analyses the vulnerability of Tornado Cash, and performs experimental analysis on the transaction set.

In summary, the contributions of this paper include:

- We first formally analyze the correlation of transactions on the Tornado Cash coin mixing service and systematically summarize the behavior patterns of users in Tornado.
- We propose three heuristic clustering rules to achieve address correlation for Tornado coin mixing transactions based on the time interval features behind the proposed two types of transaction patterns.
- We perform the experimental analysis on the real-world transaction dataset in Tornado Cash. The results prove the feasibility and effectiveness of the proposed heuristic clustering rules.

The remainder of the paper is organized as follows: In Sect. 2, we briefly review the related work on Ethereum address clustering. Section 3 presents the background knowledge of the proposed approach. Section 4 gives formal definitions of transactions and summarises transaction patterns based on the analytics of data. Section 5 proposes heuristic clustering rules and discusses the experiment results. Section 6 concludes this paper and describes the future work.

2 Related Work

In recent years, researchers pay more and more attention to analyzing Ethereum privacy. However, existing analyses are still limited and mainly explore the privacy of Ethereum users in terms of address correlation.

In general, address correlation methods on Ethereum involve two major categories. One is using machine learning and node embedding methods to cluster transaction behaviour patterns or user accounts with similar characteristics. Sun et al. [13] first applied the node embedding algorithm to the clustering of Ethereum accounts. Hu et al. [7] designed a transaction-based classification detection method for Ethereum smart contracts by summarizing contract transaction behavior patterns. Bhargavi et al. [2] analyzed the Ethereum transaction information to infer the behaviour characteristics in supervised and unsupervised environments.

¹ <https://duneanalytics.com/poma/tornado-cash-1>.

Another category is to use heuristic or graph-based clustering algorithms to link addresses that participated in certain transactions. Chan et al. [4] first explored the feasibility of using graph analysis to link Ethereum addresses. Chen et al. [6] created the finding entity algorithm between tokens on graph analysis. Chen et al. [5] analyzed the contract graph to cluster multiple smart contract accounts controlled by the same entity using weakly connected components. Victor [14] proposed heuristic address clustering rules based on users' behaviors in airdrops, ICOs and exchanges.

The existing work on analyzing the vulnerability of coin mixing service on Ethereum is Béres et al. [1]. They obtained multiple ground truth sets through heuristic rules for coin mixing service to evaluate other clustering algorithms. Nevertheless, in actual transactions, the same custom-set gas prices where the last 9 digits are non-zero are generally initiated by the same addresses. Hence, their proposed rules based on gas prices are not of practical significance.

Based on the above investigation, there is no systematic study of the address clustering that takes coin mixing services on Ethereum into account.

3 Preliminaries

This section introduces the basics of Tornado Cash and outlines the coin mixing principles and processes in Tornado Cash.

3.1 Basics of Tornado Cash

Tornado Cash is a kind of smart contract in Ethereum that uses Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARK) to achieve the unlinkability between addresses that belong to the same users, and protect their privacy in a trustless manner. This paper mainly takes ETH as an example to analyse the Ethereum transactions on Tornado Cash. To avoid address linking by unique value characteristics, Tornado deployed four smart contracts with different denominations to implement the coin mixing services of the fixed value. The detail information is shown in Table 1.

Table 1. Tornado Cash about ETH.

$N.ETH^a$	$SC_{N.ETH}^b$	Created time (+UTC)
0.1	12D66f87A04A9E220743712cE6d9bB1B5616B8Fc	2019-12-16 19:08:43
1	47CE0C6eD5B0Ce3d3A51fdb1C52DC66a7c3c2936	2019-12-16 22:17:53
10	910Cbd523D972eb0a6f4cAe4618aD62622b39DbF	2019-12-16 22:46:55
100	A160cdAB225685dA1d56aa342Ad8841c3b53f291	2019-12-25 18:02:56

^a $N.ETH$ represents the denominations of the Tornado smart contract.

^b $SC_{N.ETH}$ represents the Tornado contract addresses.

3.2 Coin Mixing Process in Tornado Cash

Users are required to complete the coin mixing in two steps: deposit and withdraw. As shown in Fig. 1, the dotted line denotes the contract invocation, and the solid line denotes the funds flow.

Fig. 1. The process of Tornado Cash coin mixing contract. (Color figure online)

The user uses his address *a* to invoke the contract $SC_{N.ETH}$ and create an $N.ETH$ deposit transaction, then he will obtain a deposit note. After a period of time, the user utilizes the previous deposit note to create an $N.ETH$ withdraw transaction, thus the contract returns the previously deposited cryptocurrency to address *b*. The refund address can be any address the user wants to remit funds to, or even an unused address that was originally unfunded. This kind of unused addresses does not have enough Ether to cover the gas fee, in this case, the user can send a request to relayer *r* with the required parameters, relayer then create a withdraw transaction upon the request as shown by the red connecting line in Fig. 2. The relayer *r* uses a fraction of f to pay the gas fee. When the contract validates the parameters in the withdrawal transaction successfully, it sends $(N - f)$ Ether to address *b* and sends f to Relayer *r*.

Because of the zero-knowledge property in zk-SNARK, Tornado guaranteed that the two transfers are completely independent. Furthermore, a deposit transaction corresponds to only one withdrawal transaction.

4 Analysis of Tornado Cash

The overview of our methodology architecture is shown in Fig. 2, which is divided into three steps: data acquisition, data analysis and cluster, and data presentation. Data acquisition includes the acquisition of related Ethereum transactions, the decoding of individual fields and the deletion of useless fields; data analysis includes statistical analysis of transaction data followed with the experimentation using proposed clustering rules; data presentation part presents the address clustering results.

Fig. 2. Overview of our methodology architecture.

4.1 Definitions

This section describes our formal definition in Tornado Cash.

Definition 1 (Transaction). Transaction is represented as $\mathbf{tx} = \{\text{hash}, \text{from}, \text{to}, \text{value}, \text{input}, \text{ts}\}$, where:

- **hash** is the hash value of $\mathbf{tx}$;
- **from** is the address that creates $\mathbf{tx}$;
- **to** is the target address of the transaction $\mathbf{tx}$. Specially, in the smart contract invocation transaction, **to** is represented the address of the smart contract;
- **value** is the value of $\mathbf{tx}$;
- **input** is invoke parameter when $\mathbf{tx}$ is a smart contract invocation transaction;
- **ts** is the timestamp representing the time when $\mathbf{tx}$ was packaged on.

The set of transactions is represented as $\mathcal{TX} = \{\mathbf{tx}_1, \mathbf{tx}_2, \dots, \mathbf{tx}_n\}$.

Definition 2 (Deposit Transaction). The deposit transaction set is represented as $\mathcal{D} = \{\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_n\} \subseteq \mathcal{TX}$, $\forall \mathbf{d}_i \in \mathcal{D}$, $\mathbf{d}_i.\text{input} = (\text{commitment})$, where the **commitment** field is the parameter used for the zk-SNARK proof.

Definition 3 (Withdraw Transaction). The withdraw transaction set is represented as $\mathcal{W} = \{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_n\} \subseteq \mathcal{TX}$, $\forall \mathbf{w}_i \in \mathcal{W}$, $\mathbf{w}_i.\text{input} = (\text{proof}, \text{nullifierHash}, \text{recipient}, \text{relayer}, \text{fee}, \text{refund})$, where **proof**, **nullifierHash** are the parameters used for the zk-SNARK, **recipient** is the target address for receiving the withdrawal funds, **relayer** is the address of the relayer, **fee** is the transaction fee given to the relayer, and **refund** is the parameter related to the refund.

- if $\mathbf{w}.\text{input}.\text{fee} \neq 0$ and $\mathbf{w}.\text{input}.\text{recipient} \neq \mathbf{w}.\text{from}$, then $\mathbf{w}$ is a withdraw transaction using the relayer, and the true withdraw target address is $\mathbf{w}.\text{input}.\text{recipient}$.
- if $\mathbf{w}.\text{input}.\text{fee} = 0$ and $\mathbf{w}.\text{input}.\text{recipient} = \mathbf{w}.\text{from}$, then $\mathbf{w}$ is a withdraw transaction not using the relayer.

4.2 Data Acquisition

Transactions data related to the four ETH denominations of the Tornado Cash coin mixer contracts are obtained by the Etherscan API².

We use the Error field to classify the transaction into **Success Transaction** and **Error Transaction**. The **Error Transaction** is categorized into *Out of gas* and *Reverted* according to the error type. For the **Success Transaction**, we decoded the input field using Contract ABI provided by Etherscan, and categorized transactions into *Deposit*, *Withdraw* and *Other*. On the basis of the above classification, we further removed the useless fields and stored the transactions in the form of *tx* associate with *type*.

Table 2 shows the categories and the corresponding number of transactions for each denomination of Ether in Tornado mixer after our processing since the deployment of the Tornado Cash in 2019 as of May 17, 2021. In addition to deposit and withdrawal transactions for coin mixing, there are a small number of failed transactions, as well as individual transactions related to contract creation, etc.

Table 2. Tornado Cash transactions details about ETH.

Value (ETH)	Success transaction			Error transaction		Total	Anonymity set size
	<i>Deposit</i>	<i>Withdraw</i>	<i>Others</i>	<i>Out of gas</i>	<i>Reverted</i>		
0.1	5122	4159	14	205	114	9613	6079
1	6299	5694	2	90	240	12325	6156
10	6988	6572	2	29	184	13775	5655
100	4288	3919	1	13	80	8301	2765
All	22697	20344	19	337	618	44014	16661

Figure 3 shows the percentage of transactions for each denomination of Tornado mixer. As is illustrated in Fig. 3, the 10ETH mixer has the largest number of transactions, while the 100ETH mixer has the least.

4.3 Transaction Patterns

From the analysis of the transaction data, we found a special phenomenon that several transactions were created within a short period of time δ (called a small transaction set $\mathcal{TX}_i$). The time interval Δ between different $\mathcal{TX}_i$ will be much larger than the time interval δ in $\mathcal{TX}_i$ internally. Besides, during the data processing, we discovered that several users in Tornado used the same addresses to deposit and withdraw. Compared their transaction time intervals, the transactions in a small set are created by the same users. In other words, a user tends to create transactions within a short period of time. This phenomenon infers that diverse addresses of transactions within the small transaction set may be controlled by the same user.

² <http://api.etherscan.io/api>.

Fig. 3. Proportion of Tornado mixer transactions by denomination.

From our analysis, we summarize two kinds of the transaction behavior patterns of users, as defined below:

Definition 4 (Single Deposit-Withdraw Coin Mixing Pattern). *The user initiates a deposit transaction $\mathbf{d}$ of $N.ETH$ using address a . After an interval δ , the user creates a withdraw transaction $\mathbf{w}$ of $N.ETH$ using address b . The above pattern is defined as **pattern I** : $\langle \mathbf{d}, \mathbf{w}, \delta \rangle$, where $\delta = \mathbf{w.ts} - \mathbf{d.ts}$, and $\mathbf{d}$, $\mathbf{w}$ satisfy the following conditions:*

- $\mathbf{d.from} = a$;
- $\mathbf{w.input.recipient} = b$;
- $\mathbf{d.to} = \mathbf{w.to}$.

Definition 5 (Multi-Deposit and Multi-Withdraw Coin Mixing Pattern). *The user creates n ($n \geq 2$) deposit transactions $\mathcal{D} = \{\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_n\}$ of $N.ETH$ using the address set $\mathcal{A} = \{a_1, a_2, \dots, a_n\}$. After an interval Δ , the user create n withdraw transactions $\mathcal{W} = \{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_n\}$ of $N.ETH$ using address set $\mathcal{B} = \{b_1, b_2, \dots, b_n\}$. The above pattern is defined as **pattern II**: $\langle \delta_d, \mathcal{D}, \delta_w, \mathcal{W}, \Delta, n \rangle$, where $\delta_d = \max\{\mathbf{d}_{i+1.ts} - \mathbf{d}_i.ts \mid \mathbf{d}_i, \mathbf{d}_{i+1} \in \mathcal{D}\}$, $\delta_w = \max\{\mathbf{w}_{i+1.ts} - \mathbf{w}_i.ts \mid \mathbf{w}_i, \mathbf{w}_{i+1} \in \mathcal{W}\}$, $\Delta = \mathbf{w}_1.ts - \mathbf{d}_n.ts$, and transactions in $\mathcal{D}$, $\mathcal{W}$ satisfy the following conditions:*

- $\forall \mathbf{d}_i \in \mathcal{D}, \mathbf{d}_i.from \in \mathcal{A}$;
- $\forall \mathbf{w}_i \in \mathcal{W}, \mathbf{w}_i.input.recipient \in \mathcal{B}$;
- $\forall \mathbf{tx}_i, \mathbf{tx}_j \in \mathcal{D} \cup \mathcal{W}$, and $i \neq j$, $\mathbf{tx}_i.to = \mathbf{tx}_j.to$.

Based on the above summaries of transaction patterns, and the fact that some users have the mentality of relying entirely on the tool. They think that Tornado Cash can achieve the address unlinkability without any consideration, therefore eager to withdraw immediately after deposits. We can analyze the time interval among coin mixer transactions to further link the diverse addresses owned by the same user.

5 Heuristic Cluster Rules

In this section, we perform the statistical analysis to propose the heuristic cluster rules linking the addresses belonging to the same user.

5.1 Heuristics

Figure 4 is the statistical results of the time intervals for previous identified users who used the same address to initiate coin mixing transactions based on the transaction patterns defined in Sect. 4.

Fig. 4. The interval time statistic based on the transaction patterns.

As shown in the green block in Fig. 4, the time interval δ in **pattern I**, where **d** and **w** are created by the same user, is basically no more than 180s. It indicates that there is a subset of users in the Tornado mixer prefer to deposit and withdraw within a short period of time. This interval δ is much smaller than the average 2h transaction interval that is common in the Tornado transaction set.

Heuristic 1: For **pattern I** $\langle \mathbf{d}, \mathbf{w}, \delta \rangle$, if $\delta \leq 180\text{s}$, the addresses in the transaction pairs $\{\mathbf{w.inpit.recipient}, \mathbf{d.from}\}$ belong to the same user.

Also, when analyzing the time interval in **pattern I**, it was found that the same user may create multiple tuples $\langle \mathbf{d}, \mathbf{w}, \delta \rangle$ successively. Their interval distribution is shown in the yellow box in Fig. 4.

δ_{dw} denotes the maximum δ_i in the multiple transaction tuples $\{\langle \mathbf{d}_1, \mathbf{w}_1, \delta_1 \rangle, \langle \mathbf{d}_2, \mathbf{w}_2, \delta_2 \rangle, \dots, \langle \mathbf{d}_n, \mathbf{w}_n, \delta_n \rangle\}$, which is presented as the pure yellow block in Fig. 4; δ_{wd} denotes the maximum time interval between each $\langle \mathbf{d}, \mathbf{w}, \delta \rangle$ tuples, which is presented as the dotted yellow block in Fig. 4.

It can be seen that the δ_{dw}, δ_{wd} of multiple $\langle \mathbf{d}, \mathbf{w}, \delta \rangle$ tuples are larger than the time interval δ of a single $\langle \mathbf{d}, \mathbf{w}, \delta \rangle$ tuple, about 20 min.

Heuristic 2: For multiple single deposit and withdraw coin mixing pairs $\{<\mathbf{d}_1, \mathbf{w}_1, \delta_{dw1}>, <\mathbf{d}_2, \mathbf{w}_2, \delta_{dw2}>, \dots, <\mathbf{d}_n, \mathbf{w}_n, \delta_{dwn}>\}$, when $n \geq 2$, if $\forall \{\mathbf{d}_i, \mathbf{d}_{i+1}\} \subseteq \{\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_n\}, \{\mathbf{w}_i, \mathbf{w}_{i+1}\} \subseteq \{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_n\}$, $\delta_{wd} = \mathbf{d}_{i+1}.\mathbf{ts} - \mathbf{w}_i.\mathbf{ts}$, satisfying one of the following conditions:

- $\mathbf{d}_i.\mathbf{from} = \mathbf{d}_{i+1}.\mathbf{from}$, and $\delta_{dwi}, \delta_{dw(i+1)} \leq 20 \text{ min}$, $\delta_{wd} > 0$.
- $\mathbf{w}_i.\mathbf{input.recipient} = \mathbf{w}_{i+1}.\mathbf{input.recipient}$, and $\delta_{dw(i+1)} > 0$, $\delta_{wd}, \delta_{dwi} \leq 20 \text{ min}$.
- $\mathbf{d}_i.\mathbf{from} = \mathbf{d}_{i+1}.\mathbf{from}$, and $\mathbf{w}_i.\mathbf{input.recipient} = \mathbf{w}_{i+1}.\mathbf{input.recipient}$, $\delta_{wd}, \delta_{dwi}, \delta_{dw(i+1)} > 0$.

The addresses $\{\mathbf{d}_1.\mathbf{from}, \mathbf{d}_2.\mathbf{from}, \dots, \mathbf{d}_n.\mathbf{from}, \mathbf{w}_1.\mathbf{input.recipient}, \mathbf{w}_2.\mathbf{input.recipient}, \dots, \mathbf{w}_n.\mathbf{input.recipient}\}$ in these transactions belong to the same user.

The distribution of the time intervals δ_d , δ_w and Δ , for the same user in *pattern II* $<\delta_d, \mathcal{D}, \delta_w, \mathcal{W}, \Delta, n>$ is shown in the pink blocks in Fig. 4. The pink slash blocks and vertical blocks are the maximum time interval between deposit transactions δ_d and withdraw transactions δ_w , respectively; the pure pink block is the time interval Δ . It seems that users tend to create a series of $\mathbf{d}$ in $\mathcal{D}$, and a series same amount of $\mathbf{w}$ in $\mathcal{W}$, with a small δ_d and δ_w . However, the time interval Δ between $\mathcal{D}$ and $\mathcal{W}$ is generally longer and more irregular.

Heuristic 3: For *pattern II* $<\delta_d, \mathcal{D}, \delta_w, \mathcal{W}, \Delta, n>$, if satisfying the following conditions:

- $\mathbf{d}_1.\mathbf{from} = \mathbf{d}_2.\mathbf{from} = \dots = \mathbf{d}_n.\mathbf{from}$;
- $\mathbf{w}_1.\mathbf{input.recipient} = \mathbf{w}_2.\mathbf{input.recipient} = \dots = \mathbf{w}_n.\mathbf{input.recipient}$;
- $\delta_d, \delta_w \leq 10 \text{ min}$ and $\Delta \leq n * 12 \text{ h}$.

The addresses $\{\mathbf{d}_1.\mathbf{from}, \mathbf{d}_2.\mathbf{from}, \dots, \mathbf{d}_n.\mathbf{from}, \mathbf{w}_1.\mathbf{input.recipient}, \mathbf{w}_2.\mathbf{input.recipient}, \dots, \mathbf{w}_n.\mathbf{input.recipient}\}$ in the n deposit and withdraw transactions belong to the same user.

In particular, the interval threshold Δ between $\mathcal{D}$ and $\mathcal{W}$ is related to the number of transactions n . The number of transactions n is increasing while the number of *pattern II* transactions is decreasing, then the interval threshold between $\mathcal{D}$ and $\mathcal{W}$ should be raised appropriately. In the sense that the threshold Δ is proportional to the number of transactions n .

5.2 Evaluation

We implement a proof-of-concept for the proposed three clustering rules on the Tornado coin mixer and make thorough experiments to verify the effectiveness of our rules. The program is written in the Python language and run in the

Table 3. Result of Heuristic 1–3.

Value (ETH)	Heuristic 1		Heuristic 2		Heuristic 3	
	<i>userNum</i>	<i>addrNum</i>	<i>userNum</i>	<i>addrNum</i>	<i>userNum</i>	<i>addrNum</i>
0.1	527	1073	61	161	119	255
1	382	840	124	334	270	600
10	310	717	106	308	353	809
100	134	371	68	246	135	373
All	1168	2734	309	953	684	1816

Python 3.6 environment based on the Windows 10 OS, with 2.5 GHz Intel Core i5-7200U CPU and 12 GB RAM.

The results of the experiment are shown in Table 3, where *userNum* and *addrNum* represents the number of clustered user entities and the number of addresses clustered in total, respectively.

As can be seen in Table 3, *Heuristic 1* has the largest number of associative clusters and clustered addresses, while the denomination 0.1ETH having the highest number, reaching 1073. After combining all the clustered results, we eventually obtain 2734 addresses related to 1168 user entities. In *Heuristic 2* and *Heuristic 3*, the highest number of associations clusters for transactions is 1ETH and 10ETH. The 100ETH mixer has the highest degree of clustering in *Heuristic 2*, with an average of 3.6 addresses per user entity.

In the experiments, we notice a particular phenomenon that the destination address **w.input.recipient** $\neq$ **w.from**, and **w.input.fee** = 0. If the relayer forwards the withdraw transaction, it is unreasonable for him to pay gas fee in advance without any forwarding fee. Thus we can infer that the two addresses are likely to be controlled by the same entity. There are a total of 95 user entities with the above case in four mixers, containing 566 related addresses.

The experiment reveals the fact that the users' behaviors hinder the achievement of their desired privacy protection. Indeed, a large proportion of users think using coin mixing tools can unconditionally protect their privacy. Unfortunately, the short time interval of depositing/withdrawing the coins exposes the users' transaction patterns that leaks the linkability of addresses they controlled. Therefore, we suggest users avoid immediately withdrawing operations after depositing their funds and prevent multiple deposits & withdrawals of the same size with one address for better privacy concerns.

6 Conclusion and Future Work

This paper presents the first systematic analysis of Tornado Cash on privacy issues. A macro analysis of the transaction in the Tornado Cash ETH coin mixer is performed. Based on the transaction time interval, two transaction patterns are formalized and three heuristic address clustering rules are proposed. The experimental results indicate that the presented methodology can reveal the address linkability in the Tornado Cash ETH coin mixer.

In future work, we can also apply the proposed methodology to other tokens with different transaction patterns.

References

1. Béres, F., Seres, I.A., Benczúr, A.A., Quinyne-Collins, M.: Blockchain is watching you: Profiling and deanonymizing ethereum users. arXiv preprint [arXiv:2005.14051](https://arxiv.org/abs/2005.14051) (2020)
2. Bhargavi, M., Katti, S.M., Shilpa, M., Kulkarni, V.P., Prasad, S.: Transactional data analytics for inferring behavioural traits in Ethereum blockchain network. In: 2020 IEEE 16th International Conference on Intelligent Computer Communication and Processing (ICCP), pp. 485–490. IEEE (2020)
3. Bünz, B., Agrawal, S., Zamani, M., Boneh, D.: Zether: towards privacy in a smart contract world. In: Bonneau, J., Heninger, N. (eds.) FC 2020. LNCS, vol. 12059, pp. 423–443. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-51280-4_23
4. Chan, W., Olmsted, A.: Ethereum transaction graph analysis. In: 2017 12th International Conference for Internet Technology and Secured Transactions (ICITST), pp. 498–500. IEEE (2017)
5. Chen, T., et al.: Understanding Ethereum via graph analysis. ACM Trans. Internet Technol. (TOIT) **20**(2), 1–32 (2020)
6. Chen, W., Zhang, T., Chen, Z., Zheng, Z., Lu, Y.: Traveling the token world: a graph analysis of Ethereum ERC20 token ecosystem. In: Proceedings of The Web Conference 2020, pp. 1411–1421 (2020)
7. Hu, T., et al.: Transaction-based classification and detection approach for ethereum smart contract. Inf. Process. Manag. **58**(2), 102462 (2021)
8. Le, D.V., Gervais, A.: Amr: Autonomous coin mixer with privacy preserving reward distribution. arXiv preprint [arXiv:2010.01056](https://arxiv.org/abs/2010.01056) (2020)
9. Meiklejohn, S., Mercer, R.: Möbius: trustless tumbling for transaction privacy. Proc. Priv. Enhanc. Technol. **2018**(2), 105–121 (2018)
10. Rondelet, A., Zajac, M.: Zeth: On integrating zerocash on ethereum. arXiv preprint [arXiv:1904.00905](https://arxiv.org/abs/1904.00905) (2019)
11. Seres, I.A., Nagy, D.A., Buckland, C., Burcsi, P.: Mixeth: efficient, trustless coin mixing service for ethereum. In: International Conference on Blockchain Economics, Security and Protocols (Tokenomics 2019). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2019)
12. Shlomovits, O., Seres, I.A.: Sharelock: mixing for cryptocurrencies from multiparty ecDSA. IACR Cryptol. ePrint Arch. **2019**, 563 (2019)
13. Sun, H., Ruan, N., Liu, H.: Ethereum analysis via node clustering. In: Liu, J.K., Huang, X. (eds.) NSS 2019. LNCS, vol. 11928, pp. 114–129. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-36938-5_7
14. Victor, F.: Address clustering heuristics for ethereum. In: Bonneau, J., Heninger, N. (eds.) FC 2020. LNCS, vol. 12059, pp. 617–633. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-51280-4_33
15. Zhang, X.: Mixing strategies in cryptocurrencies and an alternative implementation. arXiv preprint [arXiv:2010.01670](https://arxiv.org/abs/2010.01670) (2020)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.