Breaking the Anonymity of Ethereum Mixing Services Using Graph Feature Learning

Hanbiao Du, Zheng Che, Meng Shen, *Member, IEEE*, Liehuang Zhu, *Senior Member, IEEE*, and Jiankun Hu, *Senior Member, IEEE*

***Abstract*— With the property of helping users further enhance the anonymity of transactions, mixing services in blockchain have gained wide popularity in recent years. However, the strong untraceability offered by mixing services has led to the abuse of them by criminals for money laundering and committing fraud. These illegal actions pose significant threats to the blockchain ecosystem and financial order. In this paper, we focus on the problem of correlating the addresses of mixing transactions in Tornado Cash, a widely-used mixing service on Ethereum. We propose a graph neural network framework named MixBroker, which aims to break the anonymity of Tornado Cash by correlate mixing addresses from the perspective of node-pair link prediction. Specifically, we construct a Mixing Interaction Graph (MIG) using raw Ethereum mixing transaction data that can be used for subsequent analysis. To better represent the properties of mixing account nodes, we extract features from account nodes in the MIG from multiple perspectives. Furthermore, we design a GNN-based link prediction mechanism to serve as the backbone of MixBroker. This mechanism captures the interconnected nature of nodes within the MIG and calculates the probability of correlation between account nodes through node embeddings. In addition, to solve the problem of lacking ground-truth, we collect a large number of real mixing transactions of Ethereum in Tornado Cash and construct a ground-truth dataset by combining the principles of Ethereum Name Service (ENS). We conduct extensive experiments on the datasets, and the results demonstrate that MixBroker has a superior performance over other state-of-the-art methods on the address correlation problem in Ethereum mixing transactions.**

***Index Terms*— Blockchain, ethereum, mixing services, deep learning, graph neural networks.**

Manuscript received 18 April 2023; revised 23 August 2023 and 7 October 2023; accepted 7 October 2023. Date of publication 23 October 2023; date of current version 22 November 2023. This work was supported in part by the National Key Research and Development Program of China under Grant 2020YFB1006100; in part by the China National Funds for Excellent Young Scientists under Grant 62222201; in part by the Beijing Nova Program under Grant Z201100006820006 and Grant 20220484174; in part by the NSFC Project under Grant 61972039; in part by the Beijing Natural Science Foundation under Grant M23020, Grant L222098, and Grant 7232041; in part by the ARC Discovery under Grant DP190103660 and Grant DP200103207; and in part by the ARC Linkage under Grant LP180100663. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Paolo Gasti. *(Hanbiao Du and Zheng Che contributed equally to this work.) (Corresponding author: Meng Shen.)*

Hanbiao Du, Meng Shen, and Liehuang Zhu are with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China (e-mail: duhanbiao@bit.edu.cn; shenmeng@bit.edu.cn; liehuangz@bit.edu.cn).

Zheng Che is with the School of Computer Science, Beijing Institute of Technology, Beijing 100081, China (e-mail: chezheng@bit.edu.cn).

Jiankun Hu is with the School of Engineering and IT, University of New South Wales, Canberra, ACT 2610, Australia (e-mail: J.Hu@adfa.edu.au).

Digital Object Identifier 10.1109/TIFS.2023.3326984

I. Introduction

ETHEREUM is the first public blockchain platform with a prosperous ecosystem that leverages smart contracts in recent years [1]. However, the pseudonymity provided by Ethereum is vulnerable to existing attacks [2], [3], [4], [5], which has led to the development of mixing services. Mixing services can further enhance the anonymity of transactions by mixing the funds of multiple users and creating a random mapping relationship between existing and new accounts, making a transaction consists of multiple unlinkable transaction senders and receivers. As a promising approach to protect user privacy on Ethereum, mixing services have gained widespread attention in both academia [6], [7] and industry [8]. Tornado Cash [9] is a popular mixing service on Ethereum, which incorporates smart contracts to hide the relationship between senders and receivers of the original transactions. According to statistics from blockchain sites [10], as of August 2023, nearly 3.8 million ETH have been obfuscated through Tornado Cash.

Fig. 1 shows a simple example of mixing services provided by Tornado Cash, where a user Alice aims to secretly transfer funds from her address $Addr_1$ to another address she owns. Specifically, Alice first sends a locally-generated $note$ to the mixing contracts (Step 1) and then initializes a deposit transaction Tx_1 with the $Addr_1$ to deposit funds into the mixing pool (Step 2). Alice uses the $note$ to prove her deposit funds to the mixing contracts (Step 3) and creates a withdrawal transaction Tx_2 with another address $Addr_2$ she owns (Step 4a). If she generates a new account that does not have enough balance to pay the transaction fee, it is possible to create a withdrawal transaction through a $Relayer$ transaction (Step 4b), which a kind of optional service provided by Tornado Cash (more details will be introduced in Section II). The mixing contracts can hide the relationship between the actual sender (i.e., $Addr_1$) and receiver (i.e., $Addr_2$ or $Addr_3$).

Although effective in anonymity enhancement, Tornado Cash has also been abused for cybercrimes such as money laundering. More than $7 billion worth of cryptocurrency was laundered through Tornado Cash [11]. In addition, cryptocurrency exchange platform Crypto.com was stolen in January 2022, with hackers stealing 4,830 ETH and sending them to Tornado Cash to evade tracking [12]. The untraceability provided by Tornado Cash poses a severe threat to the blockchain ecosystem and financial order, as it is challenging to correlate the sender and receiver addresses of illegal transactions.

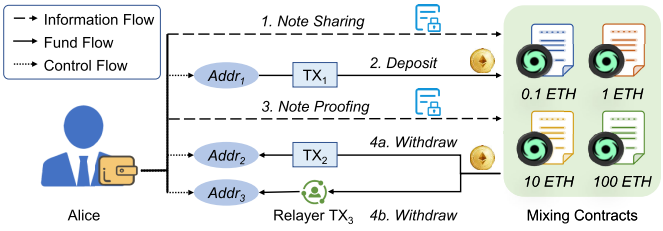


Fig. 1. The process of user interaction with Tornado Cash.

Therefore, there is an urgent need to break the anonymity of Tornado Cash by correlating the mixing transaction addresses.

Recent studies proposed several methods to find the relationship between senders and receivers of mixing transactions based on specific transaction features, such as transaction behaviors [13] and transaction amount [14]. However, Tornado Cash restricts the transaction output to fixed amounts (e.g., 0.1 ETH, 1 ETH, 10 ETH, 100 ETH), observers cannot reveal the linkage of transaction addresses by analyzing only transaction features, leading to low accuracy of existing methods. Besides, efficiency is also of great importance in address correlation. Existing methods based on heuristics [15], [16] need to traverse the entire transaction data, resulting in high time complexity. The huge and fast-growing data of mixing transactions in Ethereum leads to high time complexity of existing methods.

To tackle these challenges, we propose MixBroker, a mixing address correlation framework using graph feature learning. In order to abstract the mixing behavior of users, we construct an information-rich graph structure named Mixing Interaction Graph (MIG) to represent the transaction features and their interconnected topology structure of user accounts in Tornado Cash. The MIG has the advantage of preserving information related to the original transactions, such as the number of transactions, timestamps, gas price, and allows further extraction of deeper features. In addition, the information of interconnected transaction topology structure has been incorporated. With MIG, we successfully convert the mixing address correlation problem into a graph node-pair link prediction task. In order to mine the hidden transaction features including their topology information in MIG, we design a GNN-based prediction model, which can automatically extract deeper neighborhood features from MIG, and create links between accounts by mapping them to different representations in embedding space. MixBroker outperforms the state-of-the-art methods in correlating mixing addresses of Tornado Cash.

We summarize the main contributions as follows:

- We propose Mixing Interaction Graph (MIG) to represent each mixing transaction including its topology information, where vertices in an MIG represent user accounts and links represent mixing transactions. MIG can overcome the interference of super-nodes in the mixing pool and reveal the interaction between user accounts participating in mixing transactions.
- We design MixBroker, a powerful GNN-based mixing transaction address correlation framework integrating account feature extraction and graph feature learning. It maps MIG to different representations in GNN embedding space, extracting richer neighborhood information, which allows for accurate and efficient correlation.
- We build a Tornado Cash ground-truth dataset to reveal the connection between user deposit and withdrawal addresses. We combine user habits of utilizing Ethereum Name Service (ENS) to obtain the ground-truth dataset from over 333,700 Ethereum transactions. In addition, we build four larger synthetic datasets to evaluate the performance of MixBroker to against massive amounts of data. These datasets will be released publicly, which can support the research community for further investigation.
- We evaluate the performance of the MixBroker on the ground-truth and synthetic datasets. Compared with state-of-the-art methods (i.e., Heuristics [15], Logistic Regression [17], Random Forest [18], RNN [19], LSTM [20], DeepWalk [21] and Node2Vec [22]), MixBroker achieves the highest correlation F1-Score on ground-truth and synthetic datasets, with up to 64.2% improvement. As the amount of data increases, the efficiency of MixBroker is also significantly improved.

To the best of our knowledge, this is the first study that tackles the mixing address correlation problem using graph feature learning techniques. The rest of this paper is organized as follows. We first introduce the background of Tornado Cash and summarize the related work in Section II. Then, we describe the process of constructing ground-truth dataset in Section III. Next, we introduce the principle of using MIG to represent mixing transactions in Section IV and propose the GNN-based framework in Section V. We evaluate the performance of MixBroker and compare it comprehensively with the state-of-the-art methods in Section VI. After a brief discussion of MixBroker in Section VII, we conclude this paper in Section VIII.

II. Background and Related Work

In this section, we first introduce the background of Tornado Cash, and then we summarize the recent achievements in undermining the anonymity of mixing transactions.

A. Tornado Cash

Tornado Cash [9] is a popular decentralized mixing service in Ethereum, which uses Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARK) to verify transactions. As depicted in Fig. 1, a typical mixing transaction in Tornado Cash involves several entities described as follows.

Mixing Contracts: The mixing pool of Tornado Cash maintains four smart contracts in different denominations, namely 0.1 ETH, 1 ETH, 10 ETH, and 100 ETH. These contracts allow users to deposit funds from one account into a mixing pool and withdraw the same amount to a different account. For instance, a user with the address $0 \times 0000000000d1921aa255cc20daec11a22b7e894d$ deposited 232 ETH, which was achieved by invoking 2 times 100 ETH, 3 times 10 ETH and 2 times 1 ETH, respectively. At the same time, these mixing contracts also maintain a list

TABLE I
SUMMARY OF EXISTING STUDIES

Categories	Goals	Refs.	Methods	Transaction Features
Mixing Service Detection	Mixing Transaction Identification	[6]	Heuristic analysis	The relationship between transaction input
		[23]	Random forests	Transaction address type and transaction amount
	Mixing Address Identification	[7]	Logistic regression	Transaction order, transaction value and transaction time
		[24]	Heuristic analysis	Degree of aggregation of transaction graph
	Mixing Process Exploration	[25]	UTXO tracing	Reference relationships between UTXO
Mixing Address Correlation	Bitcoin Mixing Service Association	[14]	Heuristic analysis	Transaction value and transaction time
	Ethereum Mixing Service Association	[15][13][16][26]	Heuristic analysis	Address reuse, gas price, transaction time
		MixBroker	Graph nerual network	Graph representation of transaction interaction features

of user-encoded locks to record the encoded notes, which is used to verify the note provided when a user withdraws his funds.

Note: A note is a long sequence of digits generated locally by a deposit user and kept by the user without public disclosure. When a user deposits funds, he sends the encoded note to the mixing contract, which can be used as a proof of deposit. When the user initializes a withdrawal transaction, he must first give the note to the mixing contract for zero-knowledge proof. Upon receipt of the note, the mixing contract will perform the following checks: 1) whether the note exists in the list of user-encoded locks, 2) whether the note is the same as the deposit note, and 3) whether the note has been previously submitted. After passing the verification, the mixing contract returns the funds to the user, and the note entry is then marked as stale in the list of user-encoded locks.

Relayer: It is an independent operator that provides an optional service for Tornado Cash users to help them solve the *fee payment dilemma* (i.e., how to pay for funds withdrawals from a mixing pool while maintaining anonymity). Relayer can send a withdrawal transaction instead of the user to a new account with no ETH balance and deduct the withdrawal fee directly from the transfer amount. Specifically, the user first needs to select a Relayer to create a withdrawal transaction, and the Relayer will withdraw funds to the new address on behalf of the user. The Relayer transaction shown in Fig. 1 creates two internal transactions: one withdraws the transaction fee from mixing pool to the account of Relayer, while the other transfers the remaining funds to the account of user.

B. Summary of Existing Studies

The analysis of mixing services has received considerable attention for suspicious money flow tracing and malicious behavior analysis [27], [28], [29], [30]. We briefly review the studies on the anonymity of mixing services, which can be roughly classified into two categories, as shown in Table I.

Mixing Service Detection: The studies in this category mainly detect the transactions and addresses belonging to mixing services. Wu et al. [6] proposed a generic model for mixing services and adopted seed inputs to discover transaction outputs participating in mixing services. This method only applied to traditional mixing services that fully rely on on-chain mechanisms, such as Chipmixer [8] and Wasabi Wallet [31]. In addition to heuristics, there exists a limited corpus of literature that leveraged AI models. Wu et al. [7] constructed transaction motifs by the common behavior of mixing services and proposed a two-stage positive and unlabeled learning model based on a logistic regression classifier. However, this detection method must be based on known mixing rules and manual extraction of statistical features. Stütz et al. [23] trained a random forest classifier based on the dataset of Wasabi mixing transactions and hand-extracted transaction features to transform the problem of mixing transaction detection into a binary classification problem (i.e., Wasabi and non-Wasabi transactions). Due to the differences of structures between Bitcoin and Ethereum, these methods do not apply to the detection of Ethereum mixing services.

Mixing Address Correlation: The goal of this category is mainly to establish correlations between deposit and withdrawal addresses. Hong et al. [14] associated mixing addresses to the Helix service based on the principle that the deposits of users into mixing pools are roughly equal to their withdrawals. The analysis performed a side-channel analysis of mixing address correlation using the sensitive information of transaction amounts. Since the account model of Ethereum differs from the UTXO model of Bitcoin, Béres et al. [15] explored the address correlation method for mixing transactions under the account model. They used the last four digits of the transaction fee as a user fingerprint to establish the correlation between the sender and receiver of the mixing transaction. Based on their observations, researchers adopted various transaction information to correlate mixing addresses, including transaction time [13], transaction combination amounts [26], and address for receiving mining rewards [16].

The limitations of existing methods lie in two aspects. First, traditional heuristic methods (e.g., [15], [16], [26]) rely on individual transaction features such as transaction time and amount, necessitating pattern matching across all addresses. However, these methods struggle to achieve high efficiency when confronted with the continuously expanding volume of transaction data. Second, machine learning methods (e.g., [7], [23]) fail to effectively leverage the interdependencies among addresses, leading to diminished accuracy when analyzing mixing transactions with complex interactions (see Table X).

Our goal is to break the anonymity of Tornado Cash accurately and efficiently. By carefully observing the interaction

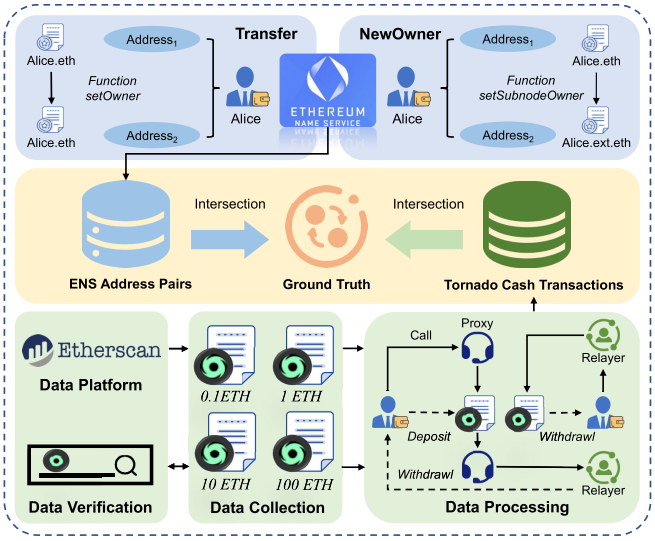


Fig. 2. The process of data preparation.

of transactions in Tornado Cash, we innovatively propose to represent mixing transactions and their topology information in graph, which naturally convert the correlation of Tornado Cash into a graph node-pair link prediction task, thus addressing the above mentioned existential challenges from a new perspective. Then, GNNs are employed to build a powerful correlation framework that maps MIGs to different representations in the embedding space and conducts transactions correlation effectively and accurately. The evaluation results show that the proposed method is superior to the state-of-the-art methods mentioned in Section VI.

III. Data Preparation

In this section, we first introduce the difficulties in constructing ground-truth dataset, and then describe the process of building ground-truth dataset in terms of data collection and preprocessing in detail.

A. Data Description

After surveying a large amount of relevant literature, we find that there is currently no publicly available dataset in Tornado Cash that can indicate whether two accounts belong to the same user. Therefore, we first need to build a real and rigorous ground-truth dataset. To achieve this goal, we should address the following challenges.

Unbalanced transaction addresses: The transaction data of Ethereum is enormous, but the addresses associated with Tornado Cash represent only a tiny fraction of Ethereum addresses. Thus, we should filter out the transaction addresses of Tornado Cash from those of Ethereum.

Address hiding mechanism: Users in Tornado Cash will call Relayer and Proxy components for withdrawal operations, resulting in the actual transaction addresses being hidden. Therefore, we should further filter out the real user addresses from the transaction addresses dataset of Tornado Cash.

Missing labeled information: The relationship between user addresses in Tornado Cash is ambiguous, leading to the lack of labeled addresses. We should build a high-quality labeled dataset, indicating the correlation among the deposit and withdrawal addresses of Tornado Cash users.

TABLE II
Ethereum Transaction Dataset

	# Internal transactions	# External transactions
0.1 ETH	42,499	9,253
1 ETH	99,021	11,957
10 ETH	93,313	13,524
100 ETH	56,066	8,151
Total	290,899	42,885

TABLE III
Information of Raw Data

Field	Notation	Definition
Txhash	T_h	Transaction hash value
Timestamp	t	Transaction completion time
From		Sender address of the transaction
To		Receiver address of the transaction
TxTo		Recipient address for derived transactions
Value_IN	w_{in}	The deposit amount of the transaction
Value_OUT	w_{out}	The withdrawal amount of the transaction
Gas Price	g	The price of the transaction fee
Method	m	Action performed
ParentTxFrom	p_v	The sender of the parent transaction
ParentTxTo	p_d	The recipient of the parent transaction

To tackle these challenges, we design a data preparation process, as shown in Fig. 2. Data preparation mainly includes obtaining Tornado Cash transactions and ENS address pairs, and then constructing the ground-truth dataset. Specifically, we first obtain the transaction records and contract call information of Tornado Cash by calling the API of Etherscan. We subsequently filter the Relayer and Proxy addresses in the transaction dataset to get a set of Tornado Cash transactions containing only user addresses. Then, we obtain ENS address pairs by requesting the ENS core contract, which is a collection that reveals the correlation of ENS between user addresses. With the two sets of data, we conduct an *intersection* to build a ground-truth dataset that reveals the relationship between addresses.

B. Data Collection

Etherscan is a blockchain browser that offers data search and analysis capabilities, along with a straightforward API for registered users to access transaction information. We utilize this API to crawl all Ethereum transaction data in Tornado Cash from the launch date until March 31, 2022. The fields included in the internal and external transaction data as well as the corresponding meanings are given in Table III. We use the transaction hash value *Txhash* to query Etherscan to get the *Input Data* field of each transaction, which can reveal the flow of funds in contract accounts for internal transactions. Querying the parameter *address_recipient* in *Input Data* field determines the actual user address receiving the funds, through which we can eliminate the confusion of Relayer address and Tornado Cash Proxy to the user address transaction behavior.

We obtain 333,784 transactions from four mixing pools in Tornado Cash, the details are shown in Table II.

Algorithm 1 Building Tornado Cash Transactions Dataset

Input: External dataset $Ex = (T_h, t, From, To, m, w_{in}, w_{out})$, internal dataset $In = (T_h, p_v, p_d, From, TxTo, w_{in}, w_{out})$
Output: Tornado Cash Transactions Dataset TC

1: Initialize T as an empty set
2: **for** $T_h \in Ex$ **do**
3: **if** $count(In.T_h) == 2$ **then**
4: Add $(T_h, t, In.tx1.TxTo, To, m)$ to TC
5: Delete T_h in In
6: **else**
7: Add $(T_h, t, Ex.From, To, m)$ to TC
8: Delete T_h in In
9: **for** $T_h \in In$ **do**
10: **if** $count(In.T_h) == 2$ **then**
11: $m = Withdraw$
12: Add $(T_h, t, TxTo, In.From, m)$ to TC
13: **else**
14: **if** $In.w_{in} == 0$ **then**
15: $m = Withdraw$
16: Add $(T_h, t, p_v, In.From, m)$ to TC
17: **if** $In.w_{out} == 0$ **then**
18: $m = Deposit$
19: Add $(T_h, t, p_v, TxTo, m)$ to TC
20: **Return** TC

Ethereum Name Service (ENS) [32] is a decentralized account name service based on Ethereum, which translates blockchain domain names (e.g., Alice.eth) to Web3.0 resources like blockchain addresses and decentralized websites. We can leverage the mapping from domain names to Ethereum addresses to discover the correlation of addresses. For example, if a user transfers the ownership of an ENS domain name from an original address to another address, these two addresses belong to the same user. We obtain 192,649 contract calls in total by accessing the ENS API.

C. Construction of Ground-Truth

Tornado Cash transaction dataset: Fig. 2 illustrates the construction of Tornado Cash transaction dataset. For ease of understanding, we present the construction method in Algorithm 1. The construction process considers the external and internal transaction dataset as the input and starts with an initialization of the Tornado Cash transactions dataset TC. Then, it traverses the T_h of the external dataset and stores transaction data into TC according to the number of occurrences of each T_h. At the same time, it removes all the T_h that have appeared in the internal dataset (lines 2-8). After this, it classifies the transactions based on the number of occurrences of each T_h and money flow in the internal dataset. If the same T_h appears twice, it is considered as a withdrawal transaction (lines 10-12). The remaining transactions are further divided into deposit and withdrawal transactions based on the inflow or outflow of money (lines 14-19).

Finally, we can obtain the dataset of Tornado Cash transactions, which contains only transactions among user addresses, eliminating the interference of Relayer and Proxy addresses. The Tornado Cash transaction dataset consists of 30,823 deposit and 44,814 withdrawal addresses in total.

ENS address pair dataset: ENS utilizes smart contracts on Ethereum to manage the registration and resolution of blockchain domain names. Users must pay a certain amount of Ethers for using an ENS domain name over a period of time. Users can also transfer the ownership of an ENS domain name from their own address to another via the *setOwner* and *setSubnodeOwner* functions [33]. We leverage these processes to obtain ENS address pair dataset, as shown in Fig. 2.

1) Rule 1: Function *setOwner* is used to reassign ownership of an ENS domain name to a new account, which triggers the *Transfer* event. We define the *Transfer* event formally as follows: given two addresses $a1$ and $a2$, if $a1$ transfers the ownership of an ENS domain name to $a2$ only once before the expiration of the name, then $\langle a1, a2\rangle = 1$.

2) Rule 2: Function *setSubnodeOwner* is used to create a new subdomain for an ENS and assign its ownership to the specified account, which triggers the *NewOwner* event. We define the *NewOwner* event formally as follows: given two addresses $a1$ and $a2$, if $a1$ has an ENS domain name and assigns a subdomain name to $a2$, then $\langle a1, a2\rangle = 1$.

We crawl all *Transfer* events of the ENS registry contract from December 2019, applying *Rule 1* to link 25,060 address pairs. We capture all *NewOwner* events when a user directly invokes the ENS registry contract, applying *Rule 2* to identify 3,551 linked address pairs. Thus, the ENS address pair dataset contains 28,611 address pairs in total.

Ground-truth dataset: With the data pre-processing steps described above, we acquire two datasets containing user addresses. One consists of Tornado Cash user addresses, while the other contains associated ENS address pairs. To obtain a ground-truth dataset of Tornado Cash deposit and withdrawal addresses, we can look for the user addresses that appear in both datasets. Therefore, we use the intersection of the two datasets as the ground-truth dataset, which consists of 103 user address pairs.

The fixed rules used in the data processing result in a relatively small dataset, as only a very small fraction of users leveraging Tornado Cash invoked the ENS contracts (i.e., *setOwner* and *setSubnodeOwner*). To obtain a larger scale ground-truth dataset, we consider the situation where users do not invoke ENS contracts and construct four synthetic datasets involving the majority of users, as described in Section VI.

IV. Graph Structure for Tornado Cash

In this section, we present Mixing Interaction Graph (MIG) as the representation of mixing transactions and their topology information in Tornado Cash and describe the method to construct MIGs from Tornado Cash raw data.

A. Mixing Transaction Formulation

In this paper, we focus on solving the transaction address correlation problem in Tornado Cash using graph feature learning techniques. Tornado Cash cuts off the explicit transaction

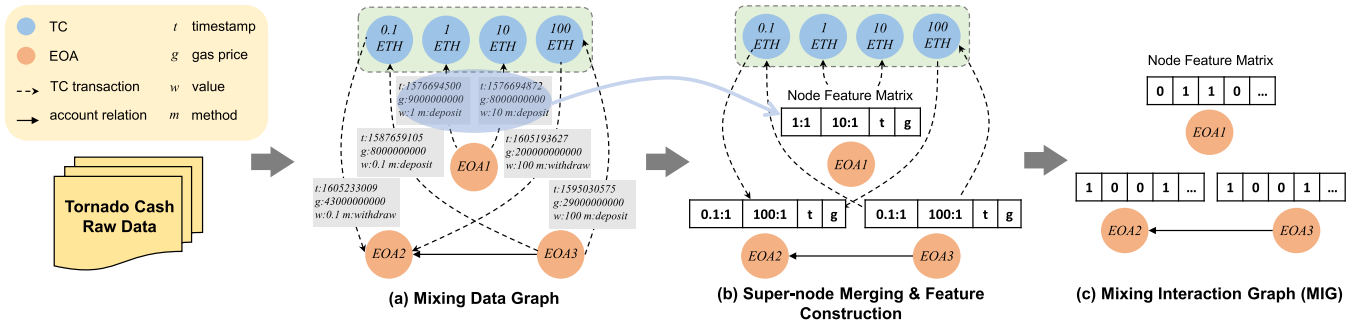


Fig. 3. Establishing mixing interaction graph through tornado cash raw data.

link between the depositor and the withdrawer, making it difficult to accurately correlate addresses by using the features extracted from a single transaction. Since user addresses interact frequently with the four smart contract addresses in the Tornado Cash mixing pool, these four addresses can be considered as super-nodes. Intuitively, this makes the process of mixing transactions become a heterogeneous interaction graph. As depicted in Fig. 3, we first convert the Tornado Cash raw data into a mixing data graph, and then combine super-node merging and feature construction to abstract the corresponding MIG, which reveals the interactions among addresses participating in mixing transactions.

Formally, we represent the transaction graph constructed from Tornado Cash raw data by a graph $G = (V, E, X)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of account nodes, $E = \{(v_i, v_j) | v_i, v_j \in V\}$ is the set of links, and $X \in \mathbb{R}^{n \times m}$is the node feature matrix. The ground-truth dataset obtained from Section III is used as the known link relationships in G. Based on the transaction graph G, the task of correlating mixing transaction addresses is to learn a function to determine the unknown linkage relationship between two accounts. The main notations used in this paper are shown in Table IV.

TABLE IV
NOTATIONS USED IN THIS PAPER

Notation	Description
G	Graph
v, V	Node, node set
E	Link set
x, X	Node feature, node feature matrix
Z	Node embedding vector in a latent space
e	Unnormalized attention scores
W	Weight parameters
α	Normalized attention scores
$\mathcal{N}(v)$	Neighborhood sampling function
h	Feature representation of node
y	Actual label of whether there is a link
$\hat{y}$	Predicted label by GNNs
$\mathcal{L}$	The loss function in GNNs

B. Mixing Interaction Graph

The initial Tornado Cash transactions are parsed to contain the raw information of mixing transactions, by which we can construct an MIG, as defined below.

Definition 1 (Mixing Data Graph): It is a directed heterogeneous graph $G = (V_{ac}, V_{tc}, E_{at}, E_{aa})$, where V_{ac} is the set of account nodes interacting with Tornado Cash and V_{tc} is the set of four smart contract nodes within the Tornado Cash mixing pool, $E_{at} = \{(v_i, v_j, t, g, w, m) | v_i \in V_{ac}, v_j \in V_{tc}\}$ is the directed link set constructed from transaction information, and $E_{aa} = \{(v_i, v_j) | v_i, v_j \in V_{ac}\}$ is the directed link set with correlation between two account nodes constructed in Section III. These four link features t, g, w, m represent *timestamp*, *gas price*, *value*, and *method* respectively, as shown in Fig. 3(a) and Table III.

The original Mixing Data Graph is a heterogeneous graph with different types of information attached to its nodes and links. All account nodes have a connection relationship with the four smart contract nodes in the mixing pool, as shown in Fig. 3(a). The heterogeneity significantly increases the difficulty of information discovery, and the presence of four smart contract nodes also confuses the connection relationship between the original account nodes. We further transform the Mixing Data Graph into a uniformly sparse MIG that can be extended to other mixing services by performing **super-node merging** and **feature construction**.

Definition 2 (Mixing Interaction Graph, MIG): It is a directed homogeneous graph $G = (V_{ac}, E_{aa}, X)$, where V_{ac} is the set of account nodes interacting with Tornado Cash, $E_{aa} = \{(v_i, v_j) | v_i, v_j \in V_{ac}\}$ is the directed link set of account nodes with correlation, and X is the node feature matrix constructed from transaction data information.

Super-node Merging: In the mixing process of Tornado Cash, all deposit and withdrawal accounts interact with smart contracts in the mixing pool, thus hiding the correlation between the real deposit accounts and withdrawal accounts. However, the existence of super-nodes will make it more difficult to extract information from Mixing Data Graph. We consider extracting information from super-nodes and merging them into account nodes in a unified manner. Fig. 3(b) shows the process of super-nodes merging, the information of invoking mixing pool contracts from deposit and withdrawal accounts will be merged as part of the account node features, where the added attributes indicate the number of interactions with 0.1, 1, 10, and 100 ETH smart contracts.

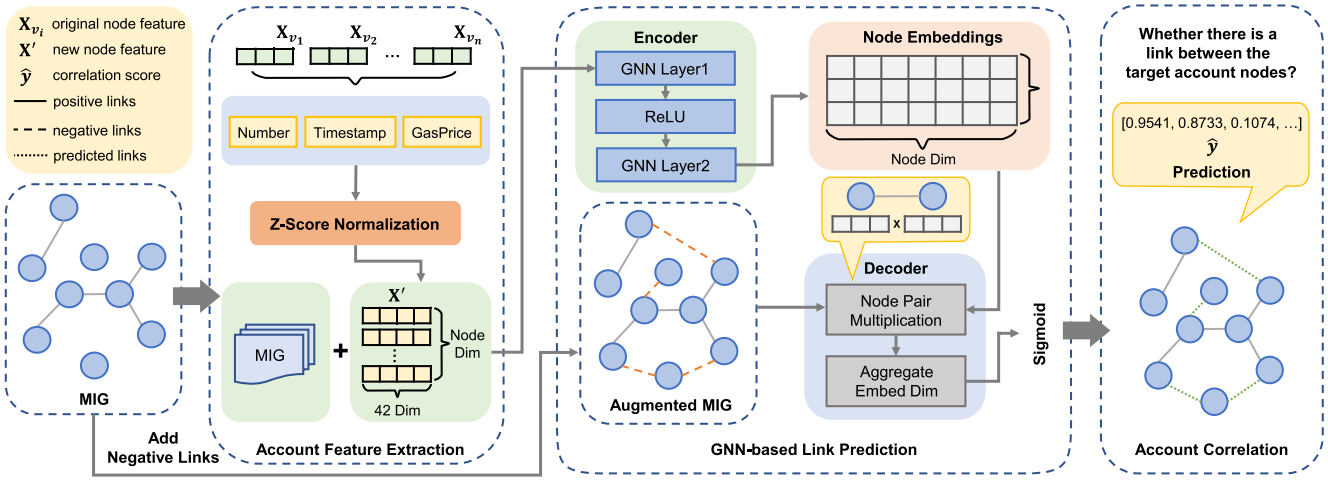


Fig. 4. The proposed MixBroker mainly contains 3 modules, including account feature extraction, GNN-based link prediction, and Account correlation.

Feature Construction: Due to the transformation of the mixing transaction address correlation problem into a node-pair link prediction task in the MIG, the interaction features of the original account node invocation of the mixing pool contracts need to be extracted and generalized after the super-node merging. The number of ground-truth dateset obtained through Section III is limited, and it is difficult to expand the mixing pool contracts invocation features into link features, so we consider transforming them into account node features. The account node features include information such as the number of interactions with the mixing pool contracts, interaction time, and interaction gas price. Specifically, we construct an account node feature matrix $X \in \mathbb{R}^{N\times M}$ to represent the interaction features of each account node with the four mixing pool contracts, where N and M denote the number of account nodes and features in MIG, respectively, as shown in Fig. 3(b). The feature matrix X can be formulated as below:

$$X = [x_1, x_2, x_3, \cdots, x_N]^\top,$$
$$x_i = [f_1, f_2, f_3, \cdots, f_M] \tag{1}$$

Note that x_i is the feature of v_i^{ac}. After the process of feature construction, we convert the heterogeneous Mixing Data Graph to a homogeneous MIG, as shown in Fig. 3(c).

In summary, the node features of MIG reflect the information about the interaction between account and mixing pool contracts in Tornado Cash.

V. The Proposed MixBroker

The correlation problem between mixing transaction addresses is transformed into a graph node-pair link prediction task using MIG. Link prediction refers to estimating the possibility of a connection between two nodes in a network, leveraging available information about the nodes and network structure. In this section, we present the implementation specifics of MixBroker, the proposed graph-based framework for correlating mixing transaction addresses.

A. Overview

As shown in the Fig. 4, MixBroker is mainly composed of the following three components: (1) an account feature extraction module, which performs deep feature extraction on the target account nodes in MIG to obtain more dimensional node feature information; (2) a GNN-based link prediction module, which encodes the MIGs and node feature information to obtain the vector representation of the nodes, and then uses a decoder to obtain the inner product between the vectors of node pairs; (3) an account correlation module to train the correlation prediction model between the mixing accounts. The input of MixBroker is the MIGs of different mixing transactions, and the output is a predictive score of whether there is a correlation between the target accounts. Next, we describe each part of the framework in detail.

B. Account Feature Extraction

There are some raw interaction features in the invocation of account nodes with Tornado Cash contracts. By extracting the features of these interactions, the hidden transaction features of accounts using mixing services can be effectively mined, which in turn correlates the addresses of the deposit and withdrawal users. Through the MIG defined in Section IV, the raw features of interaction with Tornado Cash contracts have been transformed into the features of account nodes. These raw features include three categories: the number of transactions with different contracts in the mixing pool, transaction timestamp, and transaction gas price, as shown in Fig. 4. In order to improve the performance and generability of the GNN model, we conduct deep feature extraction on these three categories of raw features, and finally identify the 42-dimensional account features (AFs), as shown in Table V.

After obtaining the 42-dimensional features, it is necessary to convert the original feature vector into a representation that is more suitable for model processing. This process is normalization, where the data are scaled to fall into a defined interval without changing the original data distribution, making the data comparable with each other. We use z-score

TABLE V
INTRODUCTION TO ACCOUNT FEATURES

Category	Feature ID (Number)	Description
Transaction Counts	AF1-AF15 (15)	The number of transactions with four mixing contracts
Gas Price	AF16-AF24 (9)	The maximum, minimum and average values of gas price
Timestamp	AF25-AF42 (18)	The earliest, latest time, total, maximum, minimum and average time interval

for the normalization task, calculated as:

$$z-\text{score}=\frac{X-\overline{X}}{S} \tag{2}$$

where X denotes the original data of each dimensional feature, $\overline{X}$ and S denote the mean and standard deviation, respectively. In Section VI, we experimentally evaluate the contribution of these features to the GNN model in detail.

C. GNN Link Prediction Mechanism

As a crucial component in MixBroker, the GNN-based link prediction mechanism is implemented based on Graph Auto-Encoders (GAE), as shown in Fig. 4. From the perspective of deep learning, the function of auto-encoder is to learn the representation of the input information as a learning target. GAE [34] is an auto-encoder based on GNN, which contains two parts (i.e., encoder and decoder) and can learn graph node information semi-supervised or unsupervised. By transforming the mixing transaction address correlation problem into a node-pair link prediction task in MIG, the account node embeddings in MIG need to be learned and then the links are predicted using node embeddings. Next, we present the details of encoder and decoder.

1) Encoder Architecture: The primary function of the encoder is to generate node embeddings of the input graph utilizing a two-layer graph neural network. This network maps the graph nodes into a low-dimensional vector space to obtain a latent representation of the graph nodes. Each layer of the graph neural network in the encoder plays a pivotal role in gathering neighboring node embeddings for each node. Subsequently, an aggregation function is employed to combine all the collected messages, followed by message passing through the update function. In the encoder architecture of MixBroker, we first use the GCN model as follows:

$$\begin{aligned} Z &= GCN(A,X) \\ &= \tilde{A}\text{ReLU}\left(\tilde{A}XW^0\right)W^1 \end{aligned} \tag{3}$$

where $Z\left(Z\in\mathbb{R}^{N\times F}, z_i\in\mathbb{R}^F\right)$ is the obtained node embedding vector through encoder, $\tilde{A}=D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ while $\tilde{A}$ and D denote symmetric normalized adjacency matrix and degree matrix respectively.

In addition, we extend the graph neural network model in encoder by introducing two other GNN models, namely GAT [35] and GraphSAGE [36], to cope with the task of correlating addresses of mixing transactions with different MIGs.

The graph attention layer is the key to the graph attention network (GAT), which defines the aggregation function through the attention mechanism. However, unlike previous models that care about information on links, the adjacency matrix in the graph attention network is only used to define the relevant nodes, while the calculation of correlation weights relies on the feature representations of the nodes. Compared with GCN, GAT introduces the self-attention mechanism, which assigns different weights to each node in the graph according to the characteristics of its neighbors when calculating its representation, thus expanding the model scale while making the calculation more efficient.

In detail, for account node v_i in the MIG, the attention mechanism calculates the similarity coefficient between its neighbors v_j and itself one by one:

$$e_{ij}^l=a\left(\left[W^lh_i^l\right]\left[W^lh_j^l\right]\right),\ j\in\mathcal{N}_i \tag{4}$$

where a linear transformation parameter W^l augments the features of the vertices and a maps the concatenated high-dimensional features to a real number. Subsequently, a nonlinear activation function $LeakyReLU$ and a $Softmax$ function are performed over the similarity coefficient e_{ij}^l to make the attention coefficient more comparable:

$$\alpha_{ij}^l=\frac{\exp\left(LeakyReLU\left(e_{ij}^l\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(LeakyReLU\left(e_{ik}^l\right)\right)} \tag{5}$$

After obtaining the normalized attention coefficients α_{ij}^l, the features are aggregated and updated according to the neighborhood of the target account node v_i:

$$h_i^{l+1}=\sigma\left(\sum_{j\in\mathcal{N}_i}\alpha_{ij}W^lh_j\right) \tag{6}$$

where h_i^{l+1} is the new feature of each account node v_i fused with the neighborhood information through a nonlinear activation function.

When training GCN, the entire adjacency matrix and feature matrix need to be input, and the Laplacian matrix of the entire topological graph is required for each node to perform the convolution operation. When a new node is added, to obtain the embedding of this added node requires the entire graph nodes to be involved in the computation all over again. GAT relies on the feature expressions of the nodes when calculating the weights between them, but only uses the information of one-hop neighbors without further depth for higher order information. Such processing hinders the application of the model to handle large-scale networks. Therefore, we further consider the use of GraphSAGE. Different from the traditional full graph convolution, GraphSAGE randomly samples the neighbors of nodes to control the number of neighboring nodes involved in the calculation and reduce the dependence on the graph structure. Afterwards, the sampled neighbor node information is aggregated using the aggregation function to obtain the embedding of the target node.

Specifically, we use k to represent the number of aggregators and the number of hops to visit the neighbors. For each account node v, the neighborhood of v is first sampled using the function $\mathcal{N}(v)$, and then the information of neighboring

nodes is aggregated into a single vector $h^k_{\mathcal{N}(v)}$ using the aggregation function:

$$h^k_{\mathcal{N}(v)} = \text{AGGREGATE}_k\left(h^{k-1}_u, \forall u \in \mathcal{N}(v)\right) \tag{7}$$

After aggregating the feature vectors of neighboring vertices, GraphSAGE updates the representation of the account node v by connecting the current embedding information of h^{k-1}_v with the aggregated neighborhood embedding information $h^k_{\mathcal{N}(v)}$ through a nonlinear activation function:

$$h^k_v = \delta\left(W^k \cdot \text{CONCAT}\left(h^{k-1}_v, h^k_{\mathcal{N}(v)}\right)\right) \tag{8}$$

GraphSAGE improves the flexibility and generalization ability of the model with faster convergence speed, which is suitable for handling address correlation tasks of mixing transactions on large-scale networks.

2) Decoder Architecture: The primary function of the decoder is to conduct link prediction for all links (both positive and negative links) using node embeddings. The original links present in MIG serve as positive samples, we need to introduce negative links into MIG to obtain augmented MIG. Following the approach for link prediction task in graph networks [37], we randomly select the same number of unknown links (i.e., unconnected node pairs) as negative samples according to the number of positive samples. The decoder calculates the inner product of the node embeddings for each link in the augmented MIG. It subsequently aggregates the values across the entire embedding dimension to generate the probability of the existence of each link, as illustrated in Fig. 4. The deposit address and withdrawal address of a mixing transaction should be close in the embedding space, and the node embedding vector Z can be served as the similarity of the nodes of a graph in some sense. As the similarity increases between two graph nodes computed from the inner product of vectors, the likelihood of a connection between the two nodes also increases. Mathematically, the reconstructed adjacency matrix can be defined as follows:

$$\begin{aligned} p(A|Z) &= \prod_{i=1}^{N}\prod_{j=1}^{N} p\left(A_{ij}|z_i, z_j\right) \\ p\left(A_{ij}=1|z_i, z_j\right) &= Sigmoid\left(z_i^\top z_j\right) \end{aligned} \tag{9}$$

where A_{ij} are the elements of A and $p(A|Z) \in \mathbb{R}^{N\times N}$ obtained through a logistic sigmoid function consistent with the dimension of adjacency matrix A.

D. Account Correlation

The account correlation module in MixBroker trains the link prediction module and obtains a correlation score for the target accounts, as shown in Fig. 4. We use the probabilities of the existence of the links output by the model to establish the correlation between the mixing transaction addresses. Classifying the probabilities requires a loss function to measure the difference between the predicted and true values, which in turn guides the next step of training the model. The loss function is defined as follows:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \cdot \log\sigma\left(\hat{y}_i\right) + (1-y_i)\cdot\log\left(1-\sigma\left(\hat{y}_i\right)\right)] \tag{10}$$

where $\mathcal{L}$ is the binary cross entropy loss and will first perform a sigmoid process on the predicted values.

VI. Performance Evaluation

In this section, we perform a comprehensive evaluation of MixBroker. First, we describe our experimental settings and introduce the synthetic datasets created on the basis of ground-truth. Second, we select different GNNs within MixBroker and analyze the impact of choosing different account node features on the results. Finally, we compare the experimental results of MixBroker with those of several baseline methods.

A. Environmental Settings

Environments: In our framework, the training operations are implemented by Python 3.7 and Pytorch 1.9.1. All experiments are run on a PC (Intel Core i7-9750H Six-Core CPU at 2.60GHz, Nvidia GeForce RTX 2070 GPU, 16GB RAM) to simulate the whole training process.

Datasets: As described in Section III, we obtain 333,784 Tornado Cash mixing transactions and 192,694 ENS contract calls from the relevant data platforms. After data processing, we obtain a ground-truth dataset, which contains 103 address pairs. The corresponding MIG contains 68,419 account nodes and 103 links, where each account node is associated with a 42-dimensional feature vector. The 103 links in the MIG will be used as positive samples after converting the mixing transaction address correlation task into a link prediction task. In addition, we randomly select the same number of non-existent links as negative samples for classifier training.

To obtain larger-scale datasets for performance evaluation, we consider the users that do not use ENS services and generate four larger synthetic datasets. The generation processes of synthetic datasets are as follows: 1) we select an equal number of negative samples as positive samples to train the model; 2) we choose a threshold θ for the classifier of the model to determine the predicted category of each sample in the test set; 3) when a sample is predicted as a positive instance, it is added to the synthetic dataset as a mixing address pair with correlation; 4) repeating the above steps multiple rounds and randomly selecting negative samples in each round to include more user addresses. The method described above enables the generation of synthetic datasets by leveraging the data distributions and patterns learned by the model. These synthetic datasets can be extended to users who do not utilize ENS services, thereby promoting greater data diversity.

We set the threshold θ to 0.5 to avoid introducing too much noisy data in synthetic datasets. These four synthetic datasets contain 1317, 15,031, 75,215, and 135,579 address pairs (i.e., links), respectively. Table VI shows the number of links contained in the five datasets, as well as the number of graph nodes included. It can be observed that SD4 includes

TABLE VI
DATASET DETAILS

Dataset	# Links	# Nodes included
Ground-Truth Dataset (GTD)	103	189
Synthetic Dataset 1 (SD1)	1,317	2,567
Synthetic Dataset 2 (SD2)	15,031	24,330
Synthetic Dataset 3 (SD3)	75,215	60,700
Synthetic Dataset 4 (SD4)	135,579	67,078

over 98% of all 68,419 nodes, which enables the model to learn sufficient information.

Metrics: As described in Section V, the correlation of user addresses have been converted into a binary classification problem. Following the literature [38], we resort to three typical metrics for evaluation, namely Precision, Recall and F1-Score. Precision is calculated by $(TP)/(TP+FP)$ and Recall is calculated by $(TP)/(TP+FN)$. A high Precision indicates that the model is less likely to misclassify negative samples as positive samples, while a high Recall indicates that the model does a better job of capturing samples that are truly positive samples. The F1-Score can be considered as a kind of harmonic average of model precision and recall, which is defined as follows:

$$\text{F1-Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{11}$$

Furthermore, we also employ two additional metrics, namely False Positive Rate (FPR) and False Negative Rate (FNR). FPR is calculated by $(FP)/(FP+TN)$ and FNR is calculated by $(FN)/(TP+FN)$. A low FPR signifies a reduced incidence of misclassification of negative samples by the model, while a low FNR indicates a diminished frequency of missed positive samples.

The K-fold cross-validation is used to evaluate the performance of each method. We randomly divide the datasets into K mutually exclusive and similarly sized subsets. Then, we use K-1 subsets as the training set and the remaining one as the test set each time. The final experimental results are taken as the average of the K tests. In the following tables, the bold fonts represent the optimal results of methods under each metric.

B. Analysis of MixBroker

In this subsection, we investigate the impact of hyper-parameters and features on the performance of different GNN models in MixBroker.

Evaluation on Hyper-parameters: An important step in training MixBroker is to tune the hyper-parameters. We explore three crucial hyper-parameters, namely the value of threshold θ, the proportion of training and testing data, and the ratio of positive to negative samples. We vary each of the hyper-parameters to evaluate the performance of MixBroker, and finally select the best combination of these hyper-parameters. A detailed comparison is summarized as follows.

1) We evaluate the impact of threshold θ in the range of 0.5-0.9 on the performance of MixBroker, as shown in Fig. 5.

TABLE VII
RATIO OF POSITIVE TO NEGATIVE SAMPLES

Metric	Pos:Neg	GCN	GAT	**GraphSAGE**
Precision	2:1	0.852±0.074	0.869±0.086	0.922±0.088
	1:1	**0.850±0.089**	**0.862±0.117**	**0.878±0.104***
	1:5	0.537±0.291	0.502±0.121	0.651±0.197
	1:10	0.334±0.127	0.368±0.165	0.728±0.241
	1:15	0.287±0.173	0.310±0.160	0.582±0.203
Recall	2:1	0.904±0.090	0.923±0.076	0.885±0.118
	1:1	**0.779±0.137**	**0.769±0.139**	**0.847±0.125***
	1:5	0.506±0.098	0.475±0.126	0.613±0.153
	1:10	0.396±0.163	0.453±0.162	0.432±0.212
	1:15	0.386±0.180	0.405±0.117	0.373±0.173
F1-Score	2:1	0.874±0.065	0.891±0.053	0.897±0.077
	1:1	**0.805±0.086**	**0.799±0.072**	**0.854±0.074***
	1:5	0.488±0.142	0.484±0.113	0.627±0.169
	1:10	0.341±0.093	0.375±0.090	0.507±0.176
	1:15	0.292±0.082	0.324±0.059	0.437±0.155

* The marked results imply that GraphSAGE outperforms GCN and GAT with a certain positive-to-negative sample ratio.

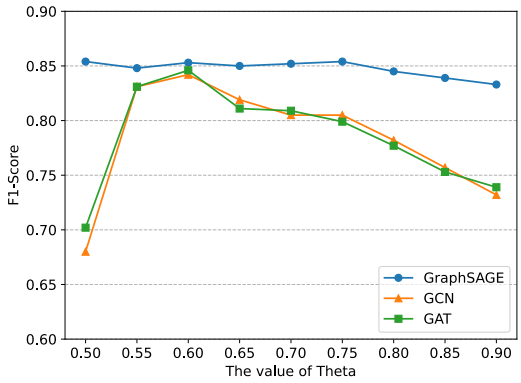


Fig. 5. The F1-Score of MixBroker with different θ values.

GraphSAGE significantly outperforms GCN and GAT, with very little variation in fluctuations, and optimal performance is obtained at 0.75. GCN and GAT obtain their best results at 0.6 and show a trend of performance degradation as θ increases.

2) Since we use K-fold cross-validation for model training and testing, we adjust the proportion of training and testing data by changing the value of K (e.g., when $K = 10$, 90% of the data is used for training and 10% for testing). According to Fig. 6, we can observe that the performance of all three models improves as the proportion of training data increases. This is because more training data allows the models to learn richer and more accurate information about the account nodes. When the proportion of training data is 90% and testing data is 10%, GraphSAGE can obtain the best performance, which is superior to GCN and GAT.

3) We set up 5 different ratios of positive to negative samples, as shown in Table VII. The model works best when the training data is twice the test data. However, reducing the number of negative samples may cause MixBroker to be more inclined to learn the features and patterns of positive samples, and less able to discriminate the negative samples.

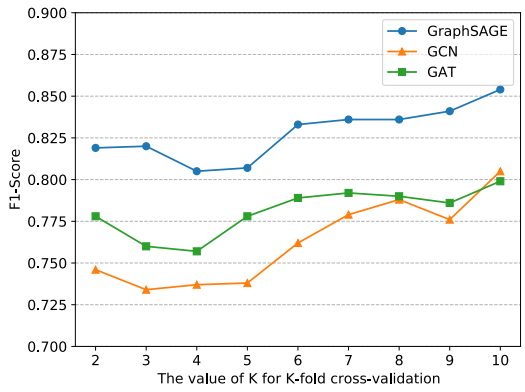


Fig. 6. The F1-Score of MixBroker with K-fold cross-validation.

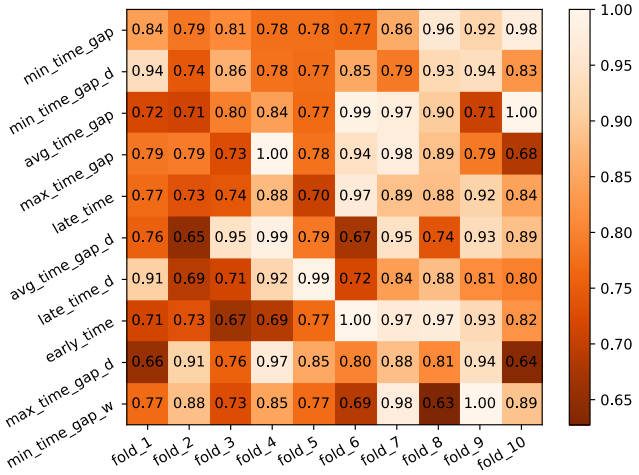


Fig. 7. The top-10 feature importance of MixBroker.

From Table VII, we can observe that with the same ratio of positive and negative samples, GraphSAGE always performs best in all three evaluation metrics. As the ratio of negative samples increases, the effectiveness of the three GNN models decreases significantly, but GraphSAGE still performs best.

GraphSAGE outperforms GCN and GAT in these experiments due to its ability to generate new node representations by sampling and aggregating features from different nodes. This approach introduces diversity in node representations, enabling GraphSAGE to better capture node-specific differences and features, as well as the local neighborhood information. GCN and GAT consider the global neighborhood information of nodes, which mainly relies on linear combinations of neighbor features. Therefore, we choose GraphSAGE as the backbone network of MixBroker hereafter. The threshold θ is set to 0.75, the same number of positive and negative samples are selected for 10-fold cross-validation.

Evaluation on Feature Importance: We evaluate the impact of selecting different dimensional account node features on the address correlation ability of mixing transactions. For GCN, GAT and GraphSAGE in the encoder module, the number of the corresponding message passing layers is 2. We set the embedding dimension, learning rate, training epoch to 16, 0.01, 100, respectively. Seven dimensions of account node features are tested in total, of which 24, 27, and 33 dimensions represent ignoring transaction timestamp features, transaction count features, and gas price features, respectively. The 42 dimensions represent all features extracted in Section V.

In addition, the 46 dimensions represent an additional 4 features related to transaction amounts, and the 100 and 235 dimensions represent the features extracted automatically using the feature generation tool. A detailed comparison is given in Table VIII and can be summarized as follows.

1) The best experimental results can be obtained for all three GNN models when 42-dimensional account node features are selected, with GraphSAGE improving the capability of 4.9%∼5.5% on F1-Score. This indicates that the transaction feature mining proposed in Section V has a positive impact on the model capability, therefore the 42-dimensional features will continue to be used in subsequent experiments.

2) Although the 100 and 235-dimensional features automatically extracted using the feature generation tool are richer in number, their experimental results on the three GNN models are also lower than our proposed 42-dimensional features due to their lack of explainability. This indicates that our extracted features are not only more explainability, but also can minimize the memory space occupied by MIG while ensuring the validity of the model.

3) The features associated with transaction timestamp have a total of 18 dimensions, and discarding these features leads to 9.9%∼17.2% reduction in the effectiveness of all three GNN models, which are the worst among the experiments. In the 10-fold cross-validation experiment, the Top-10 importance rankings of the 42-dimensional features are all timestamp features, as shown in Fig. 7. The specific descriptions of the Top-10 features are shown in Table IX. This indicates that transaction timestamp features are quite important in address correlation for mixing transactions. This observation also inspires us to get two time-focused models involved for comparison (see Section VI-C).

C. Comparison With Other Methods

Methods in comparison: To fully understand the performance of MixBroker, we employ 9 typical methods for comparison, which are briefly described as follows. To make a fair comparison, all the methods are fine tuned to achieve their best performance on the datasets used for evaluation.

- Heuristics, which correlates Tornado Cash addresses using Gas Price Fingerprint Detection (H1) and Multi-mixing Pool Detection (H2) proposed in previous related work [15]. Gas price fingerprint detection means that if a deposit transaction has the same gas price as a withdrawal transaction, and the last 9 bits of that gas price are not all zeros, it is considered that the two addresses are related to each other. Multi-mixing pool detection means that if a deposit address initiates a set of deposit transactions to several different mixing pools, and a withdrawal address initiates a set of withdrawal transactions to the same combination of multi-mixing pools within 24 hours, the two addresses are considered to be related to each other.

TABLE VIII
PERFORMANCE COMPARISON OF DIFFERENT FEATURES IN MIXBROKER

GNN Layer	Metric	24-Dim	27-Dim	33-Dim	**42-Dim**	46-Dim	100-Dim	235-Dim
GCN	Precision	0.779±0.159	0.863±0.121	0.810±0.146	0.850±0.089	0.823±0.102	**0.866±0.104**	0.798±0.130
	Recall	0.543±0.124	0.678±0.132	0.584±0.127	**0.779±0.137**	0.710±0.153	0.564±0.101	0.737±0.116
	F1-Score	0.633±0.117	0.752±0.102	0.672±0.119	**0.805±0.086**	0.755±0.112	0.675±0.073	0.755±0.080
GAT	Precision	0.807±0.155	0.859±0.133	0.815±0.147	**0.862±0.117**	0.813±0.112	0.830±0.136	0.750±0.054
	Recall	0.602±0.098	0.650±0.140	0.707±0.151	0.769±0.139	**0.797±0.122**	0.584±0.143	0.715±0.182
	F1-Score	0.677±0.065	0.732±0.117	0.745±0.108	**0.799±0.072**	0.796±0.068	0.662±0.083	0.720±0.115
GraphSAGE	Precision	0.789±0.074	0.889±0.106	0.846±0.052	**0.878±0.104***	0.848±0.081	0.826±0.109	0.766±0.129
	Recall	0.740±0.155	0.825±0.167	0.835±0.132	**0.847±0.125***	0.809±0.150	0.808±0.167	0.808±0.146
	F1-Score	0.755±0.088	0.844±0.109	0.835±0.077	**0.854±0.074***	0.823±0.107	0.806±0.110	0.781±0.125

* The marked results imply that GraphSAGE outperforms GCN and GAT under the corresponding metric.

- Logistic Regression (LR) [17], which is a generalized linear machine learning model that widely used for binary classification tasks.
- Random Forest (RF) [18], which uses multiple decison trees to binary classify the input data and determines the final output by voting or averaging.
- RNN [19], which passes information and persistent state inside the network by introducing cyclic connections to capture temporal relationships in sequential data.
- LSTM [20], which is an improvement of RNN. By introducing a gating mechanism, LSTM can control the flow and forgetting of information and is able to better maintain and utilize important contextual information when processing long sequences.
- DeepWalk [21], which captures the structural information between nodes by performing random walks on the graph and mapping them to a low-dimensional vector space.
- Node2Vec [22], which introduces two parameters based on DeepWalk: p and q. These parameters are used to control the behavior of random walk and make random walks more flexible.

These methods can be roughly categorized into 3 groups, namely Heuristics, machine learning methods (i.e., LR, RF, RNN, and LSTM), and graph embedding methods (i.e., DeepWalk and Node2Vec). For RF, we set the number of base estimators to 200. For other machine learning methods, we set the learning rate and training epoch to 0.01 and 100. For DeepWalk and Node2Vec, we set the dimension of node embeddings to 16, the length of walks to 10, the number of walks to 10, and the context size to 10. The return parameter p and in-out parameter q of Node2Vec are 0.75 and 0.75.

Evaluation on Accuracy: We investigate the effectiveness of MixBroker with the baseline methods on the five datasets. According to the results shown in Tables X and XI, we have the following main observations.

1) H2-MultiPool on GTD and H1-Gas on SD1 can obtain the highest Precision, but their performance on Recall is very poor, resulting in the worst F1-Score among all methods, as shown in Table XI. Heuristics can only utilize the isolated features manually observed from mixing transactions for address correlation, and cannot effectively use the deep feature information. Therefore, the scalability and effectiveness of Heuristics do not meet the needs of large-scale mixing transaction addresses correlation.

TABLE IX
DESCRIPTION OF TOP-10 FEATURES

Feature	Description
min_time_gap	Minimum time interval for all transactions
min_time_gap_d	Minimum time interval for all deposit transactions
avg_time_gap	Average time interval for all transactions
max_time_gap	Maximum time interval for all transactions
late_time	Time of occurrence of the last transaction
avg_time_gap_d	Average time interval for all deposit transactions
late_time_d	Time of occurrence of the last deposit transaction
early_time	Time of occurrence of the first transaction
max_time_gap_d	Maximum time interval for all deposit transactions
min_time_gap_w	Minimum time interval for all withdrawal transactions

2) MixBroker significantly outperforms other methods in terms of F1-Score on all datasets, as shown in Table X. MixBroker yields a relative improvement of 54.5%∼64.2% compared to the combined Heuristic (i.e., combining the rules of H1-Gas and H2-MultiPool). This indicates that MIG can effectively transform mixing transactions including topology information into graph structured, and carefully extracted features can improve the ability to correlate mixing addresses. MixBroker yields 1.3%∼22.5% relative improvement compared to machine learning and graph embedding methods. This indicates that MixBroker can learn both graph topology and latent features, which would make it superior to machine learning and graph embedding methods.

3) MixBroker achieves a similar FPR and a better FNR compared with other methods. This implies that MixBroker minimizes the number of missed pairs of mixing addresses with genuine correlations, leading to improved coverage. The mixing address correlation problem necessitates a thorough analysis of addresses with a focus on reducing missed instances. Therefore, MixBroker places greater emphasis on optimizing FNR to minimize the occurrence of false negatives.

4) The F1-Score of RNN and LSTM outperforms other machine learning methods, indicating that time-focused models can better utilize the timestamp features in MIG (see Section VI-B). RNN and LSTM have comparable Precision to MixBroker, but Recall is lower, resulting in inferior F1-Score. Time-focused models have higher Precision in predicting data at certain specific time steps. However, the dependencies between graph data nodes are usually nonlinear, which may result in lower Recall for time-focused models.

TABLE X
PERFORMANCE COMPARISON OF DIFFERENT METHODS

Metric	Dataset	Method						
		LR	RF	RNN	LSTM	DeepWalk	Node2Vec	**MixBroker**
Precision	GTD	0.730±0.134	0.858±0.097	0.852±0.072	0.837±0.107	0.615±0.150	0.557±0.059	**0.878±0.104**
	SD1	0.514±0.051	0.558±0.027	0.581±0.030	0.584±0.036	0.497±0.027	0.501±0.015	**0.592±0.023**
	SD2	0.500±0.012	0.641±0.008	0.631±0.008	**0.649±0.012**	0.501±0.008	0.500±0.002	0.624±0.012
	SD3	0.498±0.009	0.592±0.004	0.586±0.267	**0.607±0.179**	0.500±0.002	0.501±0.002	0.570±0.012
	SD4	0.497±0.005	**0.553±0.003**	0.544±0.024	0.538±0.029	0.501±0.001	0.499±0.002	0.545±0.006
Recall	GTD	0.674±0.157	0.662±0.210	0.720±0.131	0.788±0.142	0.815±0.212	0.760±0.194	**0.847±0.125**
	SD1	0.440±0.093	0.542±0.030	0.464±0.041	0.425±0.043	0.676±0.344	**0.699±0.297**	0.623±0.034
	SD2	0.443±0.089	0.652±0.012	0.677±0.011	0.669±0.010	0.704±0.207	0.738±0.192	**0.753±0.015**
	SD3	0.460±0.094	0.620±0.007	0.671±0.050	0.662±0.021	0.663±0.137	0.656±0.103	**0.763±0.024**
	SD4	0.462±0.088	0.571±0.005	0.680±0.073	0.706±0.068	0.634±0.089	0.684±0.102	**0.722±0.011**
F1-Score	GTD	0.683±0.077	0.724±0.132	0.775±0.083	0.799±0.066	0.666±0.104	0.629±0.079	**0.854±0.074**
	SD1	0.470±0.069	0.550±0.024	0.515±0.030	0.491±0.036	0.523±0.218	0.549±0.159	**0.607±0.025**
	SD2	0.466±0.052	0.647±0.009	0.653±0.008	0.659±0.007	0.573±0.086	0.586±0.074	**0.682±0.005**
	SD3	0.474±0.055	0.606±0.005	0.624±0.003	0.631±0.002	0.565±0.051	0.565±0.039	**0.652±0.002**
	SD4	0.475±0.050	0.562±0.003	0.602±0.012	0.608±0.007	0.557±0.036	0.574±0.037	**0.621±0.003**
FPR	GTD	0.286±0.176	0.127±0.105	**0.127±0.066**	0.176±0.131	0.592±0.245	0.627±0.252	0.135±0.131
	SD1	0.417±0.102	0.421±0.049	0.336±0.047	**0.303±0.044**	0.660±0.336	0.548±0.324	0.425±0.038
	SD2	0.441±0.082	0.365±0.011	0.397±0.013	**0.362±0.020**	0.699±0.205	0.739±0.188	0.469±0.042
	SD3	0.463±0.089	**0.427±0.005**	0.480±0.105	0.428±0.006	0.662±0.137	0.637±0.100	0.576±0.046
	SD4	0.467±0.087	**0.462±0.007**	0.577±0.123	0.615±0.129	0.632±0.090	0.687±0.100	0.529±0.027
FNR	GTD	0.326±0.158	0.338±0.162	0.280±0.131	0.212±0.142	0.185±0.212	0.240±0.146	**0.153±0.125**
	SD1	0.560±0.093	0.458±0.034	0.535±0.041	0.575±0.044	0.324±0.344	**0.301±0.218**	0.377±0.031
	SD2	0.557±0.089	0.345±0.012	0.323±0.011	0.331±0.010	0.296±0.207	0.262±0.186	**0.247±0.020**
	SD3	0.540±0.094	0.379±0.005	0.329±0.050	0.338±0.006	0.337±0.137	0.344±0.101	**0.237±0.024**
	SD4	0.538±0.088	0.429±0.005	0.320±0.073	0.294±0.069	0.366±0.089	0.316±0.102	**0.278±0.021**

TABLE XI
PERFORMANCE COMPARISON BETWEEN HEURISTICS AND MIXBROKER

Metric	Dataset	H1-Gas	H2-MultiPool	H1+H2	**MixBroker**
Precision	GTD	0.929	**1.000**	0.947	0.878
	SD1	**0.685**	0.667	0.672	0.592
	SD2	0.497	0.527	0.526	**0.624**
	SD3	0.527	0.508	0.521	**0.570**
	SD4	0.523	0.496	0.515	**0.545**
Recall	GTD	0.126	0.068	0.175	**0.847**
	SD1	0.028	0.006	0.033	**0.623**
	SD2	0.016	0.005	0.021	**0.753**
	SD3	0.016	0.004	0.019	**0.763**
	SD4	0.016	0.004	0.019	**0.722**
F1-Score	GTD	0.222	0.127	0.295	**0.854**
	SD1	0.054	0.012	0.062	**0.607**
	SD2	0.032	0.010	0.040	**0.682**
	SD3	0.031	0.008	0.037	**0.652**
	SD4	0.030	0.008	0.036	**0.621**
FPR	GTD	0.010	**0.000**	0.010	0.135
	SD1	0.013	**0.003**	0.016	0.425
	SD2	0.017	**0.005**	0.019	0.469
	SD3	0.014	**0.004**	0.018	0.576
	SD4	0.014	**0.004**	0.018	0.529
FNR	GTD	0.874	0.932	0.825	**0.153**
	SD1	0.972	0.994	0.967	**0.377**
	SD2	0.984	0.995	0.979	**0.247**
	SD3	0.984	0.996	0.981	**0.237**
	SD4	0.984	0.996	0.981	**0.278**

TABLE XII
TRAINING TIME OF DIFFERENT METHODS

Method	Training Time (s)				
	GTD	SD1	SD2	SD3	SD4
LR	9.44	88.13	964.41	4982.47	8393.84
RF	**8.89**	98.88	1161.57	6109.87	11546.31
RNN	12.13	92.81	997.05	4791.86	8424.27
LSTM	11.85	92.27	995.16	4759.82	8315.55
DeepWalk	403.77	408.13	441.92	606.61	768.27
Node2Vec	408.21	395.16	434.34	608.66	764.70
MixBroker	47.40	**49.78**	**71.67**	**170.06**	**255.52**

5) After using synthetic datasets, all methods show performance degradation in terms of five evaluation metrics. The reason for this situation is that synthetic datasets are artificially constructed and present noisy representations of account correlation in the real world. Consequently, the models acquire suboptimal representations due to the inherent noise within the synthetic datasets, thereby leading to decline in performance. However, MixBroker still outperforms the other methods on synthetic datasets, indicating that MixBroker is more robust to changes in dataset.

Evaluation on Time Efficiency: We evaluate the time cost of all the methods in two aspects: training time and inference time. Training time refers to the total time for data processing and model training based on the same set of training samples. Inference time refers to the total time for obtaining required features from the same set of testing samples and labeling them. Notice that Heuristic has no training process, we regard the time it consumes as inference time. We have the following observations through the results shown in Tables XII and XIII.

1) RF requires the shortest training time and inference time on GTD. All four machine learning methods consume less training time and inference time on GTD than Heuristic

TABLE XIII
INFERENCE TIME OF DIFFERENT METHODS

Method	Inference Time (s)				
	GTD	SD1	SD2	SD3	SD4
Heuristic	12.81	135.15	1481.42	7351.26	12664.41
LR	1.09	9.94	107.29	555.05	936.11
RF	**0.84**	**9.50**	108.26	549.81	1007.81
RNN	1.37	10.59	110.73	535.30	1010.03
LSTM	1.38	10.67	109.19	530.68	984.23
DeepWalk	40.63	42.98	54.28	107.09	155.74
Node2Vec	42.22	42.70	54.72	107.36	153.26
MixBroker	42.64	43.54	**45.99**	**57.30**	**66.51**

and graph-based methods. Heuristic method has the longest inference time across all synthetic datasets because it needs to traverse all the data for rule matching, resulting in a significant increase in time complexity.

2) The training time and inference time of both RNN and LSTM are nearly identical, and outperforms MixBroker on GTD. However, as the volume of data increases, the training time and inference time for RNN and LSTM experience significant growth. As time-focused models, the computation at each time step relies on the output from the previous time step. Optimal performance for RNN and LSTM necessitates training on the complete sequence, which contributes to a relatively slower training process.

3) As the amount of data increases, MixBroker demonstrates its superiority in terms of time efficiency. MixBroker achieves the shortest training time among SD1, SD2, SD3, and SD4, as well as the shortest inference time among SD2, SD3, and SD4. It is worth noting that on SD4, which consists of the largest data volume, MixBroker achieves a remarkable reduction in both training time (i.e., 2.9×-45×) and inference time (i.e., 2.3×-190×) over the other methods.

The evaluation on time efficiency shows that MixBroker can balance accuracy and efficiency even with rapidly growing data volumes. The localized computation of graph neural networks and parameter sharing between nodes can reduce the amount of computation and speed up the training process. When inference on large-scale graphs, GNN achieves improved inference efficiency by leveraging local neighborhood aggregation. This approach allows GNN perform feature aggregation on neighboring nodes without relying on global information from the entire graph.

D. Case Study

In this subsection, we present a real coin mixing case to demonstrate that MixBroker can find correlated addresses that cannot be discovered by fixed rules.

In September 2020, the KuCoin exchange suffered a security breach [39]. Hackers gained unauthorized access to the hot wallets of KuCoin, from which they stole a variety of cryptocurrencies. An analysis of the KuCoin theft by blockchain security firm SlowMist [40] finds that about 49.7K ETH was transferred to the 100 ETH pool of Tornado Cash. According to the report, the hackers deposited 11,500 ETH into Tornado Cash through $Addr1$,[1] and withdrew the same amount of ETH through $Addr2$[2] after a certain period of time. We utilize the addresses disclosed in the report to verify the limitations of fixed rules and to illustrate the effectiveness of MixBroker.

The fixed rules cannot establish a correlation between these two addresses due to the absence of their usage of the ENS service. In contrast, MixBroker assigns a correlation score of 1 to these two addresses, indicating a strong correlation between them. Given the identical deposit and withdrawal amounts, along with their close temporal proximity, the feature extraction module of MixBroker determines that these two addresses exhibit similar vector representations in the graph embedding matrix. Consequently, they obtain a significantly high correlation score. This finding further substantiates the superiority of MixBroker to correlate mixing addresses that cannot be discovered by the fixed rules.

VII. DISCUSSION

In this section, we discuss several issues closely related to the performance of MixBroker.

Ground-Truth Dataset: Due to the implementation of stringent constraints, the number of ground-truth dataset we construct is relatively small. In future work, more realistic datasets can be obtained by continuing to explore the transaction habits, geographic locations, IP addresses, and other information of users who leverage mixing services. Since the fixed rules of constructing ground-truth is constrained by prior knowledge, it cannot capture all the information. MixBroker can automatically learn the complex non-linear patterns and underlying features in them to cope with the demand for real-time processing and fast prediction of new data.

Dynamic Accounts: MIG is constructed based on historical transaction data. For newly added accounts in the mixing services, they can be abstracted as new nodes in MIG. MixBroker can maintain its effectiveness in a dynamic scenario by periodically updating the dataset and model parameters.

Scalability: In MixBroker, we abstract the interaction between accounts in an informative graph (i.e., MIG) and utilize GraphSAGE to learn node embeddings in an inductive way, which can occupy less memory and achieve efficient training. As shown in Tables XII and XIII, a large increase of data will not introduce significant time cost, which illustrates that MixBroker can cope with the large scale of accounts and transactions. In addition, MixBroker can employ methods such as pruning and optimized graph computation to further reduce memory consumption and training time.

Generalization: MIG is proposed to represent mixing interactions in Tornado Cash in this paper. However, it can also be applied to abstract the interactions between accounts in other types of mixing services, such as Typhoon Cash, Typhoon Network and AMR. With MIGs as input, MixBroker can further be extended to analyze and reveal the correlation among transaction accounts in these mixing services.

[1] $Addr1$: 0 × 34a17418cec67b82d08cf77a987941f99dc87c6b.
[2] $Addr2$: 0 × 82e6b31b0fe94925b9cd1473d05894c86f277398.

VIII. Conclusion

In this paper, we studied the Ethereum mixing transaction address correlation problem and broke the anonymity of Tornado Cash. We systematically analyzed the mixing process of Tornado Cash, and proposed a graph-based framework named MixBroker for mixing transaction address correlation. MixBroker could convert the original mixing transactions into MIG through super-node merging and feature construction, and input them into a GNN model for training, thus transforming the mixing transaction address correlation problem into a link prediction task. We collected a large number of Ethereum mixing transactions in Tornado Cash and produced a ground-truth dataset. The experimental results show that MixBroker can improve the correlation F1-Score by up to 64.2% over the state-of-the-art methods while maintaining time efficiency.

In future work, we plan to introduce the concept of temporal modeling into MixBroker. The temporal-GNN model allows MixBroker to mine the timing information hidden in the massive mixing transactions, further improving the effectiveness of MixBroker to correlate mixing addresses. Furthermore, we will actively explore the scalability and generalization of MixBroker on different mixing services. We think the process of creating MIG on other mixing services (e.g., Chipmixer and Typhoon Cash) will be similar with only a few changes in the feature construction process.

References

[1] V. Buterin. (2013). *A Next-Generation Smart Contract and Decentralized Application Platform*. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper

[2] J. Zhou, C. Hu, J. Chi, J. Wu, M. Shen, and Q. Xuan, "Behavior-aware account de-anonymization on Ethereum interaction graph," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 3433–3448, 2022.

[3] M. Shen, J. Zhang, L. Zhu, K. Xu, and X. Du, "Accurate decentralized application identification via encrypted traffic analysis using graph neural networks," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 2367–2380, 2021.

[4] Z. Wu, J. Liu, J. Wu, Z. Zheng, and T. Chen, "TRacer: Scalable graph-based transaction tracing for account-based blockchain trading systems," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 2609–2621, 2023.

[5] M. Shen, Y. Liu, L. Zhu, X. Du, and J. Hu, "Fine-grained webpage fingerprinting using only packet length information of encrypted traffic," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 2046–2059, 2021.

[6] L. Wu et al., "Towards understanding and demystifying Bitcoin mixing services," in *Proc. Web Conf.*, J. Leskovec, M. Grobelnik, M. Najork, J. Tang, and L. Zia, Eds. Ljubljana, Slovenia, Apr. 2021, pp. 33–44.

[7] J. Wu, J. Liu, W. Chen, H. Huang, Z. Zheng, and Y. Zhang, "Detecting mixing services via mining Bitcoin transaction network with hybrid motifs," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 4, pp. 2237–2249, Apr. 2022.

[8] *ChipMixer*. Accessed: Aug. 5, 2023. [Online]. Available: https://chipmixer.be/index.html

[9] *Tornado Cash*. Accessed: Aug. 5, 2023. [Online]. Available: https://github.com/tornadocash/tornado-core

[10] *Dune*. Accessed: Aug. 5, 2023. [Online]. Available: https://dune.com/poma/tornado-cash_1

[11] D. Yaffe-Bellany. (Aug. 2022). *Treasury Dept. Blacklists Crypto Platform Used in Money Laundering*. Accessed: Aug. 5, 2023. [Online]. Available: https://www.nytimes.com/2022/08/08/technology/treasury-blacklist-crypto-tornado-cash-laundering.html

[12] J. Crawley. (Jan. 2022). *Crypto.com Says Hackers Stole Nearly $34M From Users*. Accessed: Aug. 5, 2023. [Online]. Available: https://www.coindesk.com/business/2022/01/20/cryptocom-says-hackers-stole-nearly-34m-from-users

[13] M. Wu et al., "Tutela: An open-source tool for assessing user-privacy on Ethereum and Tornado Cash," 2022, *arXiv:2201.06811*.

[14] Y. Hong, H. Kwon, J. Lee, and J. Hur, "A practical de-mixing algorithm for Bitcoin mixing services," in *Proc. 2nd ACM Workshop Blockchains, Cryptocurrencies, Contracts*, S. V. Lokam, S. Ruj, and K. Sakurai, Eds. Incheon, South Korea, May 2018, pp. 15–20.

[15] F. Béres, I. A. Seres, A. A. Benczúr, and M. Quintyne-Collins, "Blockchain is watching you: Profiling and deanonymizing Ethereum users," in *Proc. IEEE Int. Conf. Decentralized Appl. Infrastructures (DAPPS)*, Aug. 2021, pp. 69–78.

[16] Z. Wang et al., "On how zero-knowledge proof blockchain mixers improve, and worsen user privacy," in *Proc. ACM Web Conf.*, Y. Ding, J. Tang, J. F. Sequeda, L. Aroyo, C. Castillo, and G. Houben, Eds. Austin, TX, USA, Apr. 2023, pp. 2022–2032.

[17] D. W. Hosmer, S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*, vol. 398. Hoboken, NJ, USA: Wiley, 2013.

[18] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[19] J. L. Elman, "Finding structure in time," *Cognit. Sci.*, vol. 14, no. 2, pp. 179–211, Mar. 1990.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[21] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2014, pp. 701–710.

[22] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 855–864.

[23] R. Stütz, J. Stockinger, P. Moreno-Sanchez, B. Haslhofer, and M. Maffei, "Adoption and actual privacy of decentralized CoinJoin implementations in Bitcoin," in *Proc. 4th ACM Conf. Adv. Financial Technol.*, M. Herlihy and N. Narula, Eds. Cambridge, MA, USA, Sep. 2022, pp. 254–267.

[24] M. A. Prado-Romero, C. Doerr, and A. Gago-Alonso, "Discovering Bitcoin mixing using anomaly detection," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications* (Lecture Notes in Computer Science), vol. 10657, M. Mendoza and S. A. Velastin, Eds. Valparaíso, Chile: Springer, 2018, pp. 534–541.

[25] M. Möser, R. Böhme, and D. Breuker, "An inquiry into money laundering tools in the Bitcoin ecosystem," in *Proc. APWG eCrime Researchers Summit*, San Francisco, CA, USA, Sep. 2013, pp. 1–14.

[26] Y. Tang, C. Xu, C. Zhang, Y. Wu, and L. Zhu, "Analysis of address linkability in Tornado Cash on Ethereum," in *Cyber Security*. Singapore: Springer, 2022, pp. 39–50.

[27] M. Shen, K. Ji, Z. Gao, Q. Li, L. Zhu, and K. Xu, "Subverting website fingerprinting defenses with robust traffic representation," in *Proc. 32nd USENIX Secur. Symp. (USENIX Secur.)*, 2023, pp. 607–624.

[28] H. Jin, C. Li, J. Xiao, T. Zhang, X. Dai, and B. Li, "Detecting arbitrage on Ethereum through feature fusion and positive-unlabeled learning," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 12, pp. 3660–3671, Dec. 2022.

[29] D. Das, P. Bose, N. Ruaro, C. Kruegel, and G. Vigna, "Understanding security issues in the NFT ecosystem," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, H. Yin, A. Stavrou, C. Cremers, and E. Shi, Eds. Los Angeles, CA, USA, Nov. 2022, pp. 667–681.

[30] M. Shen et al., "Machine learning-powered encrypted network traffic analysis: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 1, pp. 791–824, 1st Quart., 2023.

[31] zkSNACKs. (2017). *Official Website of Wasabi Wallet*. Accessed: Aug. 5, 2023. [Online]. Available: https://wasabiwallet.io/

[32] (2020). *Ethereum Name Service*. Accessed: Aug. 5, 2023. [Online]. Available: https://app.ens.domains/

[33] P. Xia et al., "Challenges in decentralized name management: The case of ENS," in *Proc. 22nd ACM Internet Meas. Conf.*, C. Barakat, C. Pelsser, T. A. Benson, and D. Choffnes, Eds. Nice, France, Oct. 2022, pp. 65–82.

[34] T. N. Kipf and M. Welling, "Variational graph auto-encoders," 2016, *arXiv:1611.07308*.

[35] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. 6th Int. Conf. Learn. Represent.*, 2018, pp. 1–12.

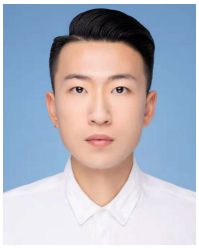
[36] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1025–1035.

[37] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–11.

[38] N. Chinchor, "MUC-4 evaluation metrics," in *Proc. 4th Conf. Message Understand. (MUC4)*, 1992, pp. 1–8.

[39] *KuCoin Security Incident*. Accessed: Oct. 2, 2023. [Online]. Available: https://www.kucoin.com/zh-hant/news/en-kucoin-ceo-livestream-recap-latest-updates-about-security-incident

[40] *SlowMist AML: Tracking funds laundered by Tornado Cash*. Accessed: Oct. 2, 2023. [Online]. Available: https://slowmist.medium.com/slowmist-aml-tracking-funds-laundered-by-tornado-cash-3a0e1f637054



Hanbiao Du received the B.Eng. degree in computer science from Shandong University, Weihai, China, in 2021. He is currently pursuing the Ph.D. degree with the School of Cyberspace Science and Technology, Beijing Institute of Technology. His research interests include blockchain data analysis and deep learning.

Zheng Che received the B.S. degree in software engineering from the Taiyuan University of Technology, Taiyuan, China, in 2020. He is currently pursuing the Ph.D. degree in computer science with the Beijing Institute of Technology, Beijing, China. His current research interests include blockchain data analysis and network security.

Meng Shen (Member, IEEE) received the B.Eng. degree in computer science from Shandong University, Jinan, China, in 2009, and the Ph.D. degree in computer science from Tsinghua University, Beijing, China, in 2014. He is currently a Professor with the Beijing Institute of Technology, Beijing. He has authored more than 50 papers in top-level journals and conferences, such as USENIX Security, ACM SIGCOMM, IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, and IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY. His research interests include data privacy and security, blockchain applications, and encrypted traffic classification. He received the Best Paper Runner-Up Award from IEEE IPCCC 2014 and IEEE/ACM IWQoS 2020. He was selected by the Beijing Nova Program 2020 and the winner of the ACM SIGCOMM China Rising Star Award 2019. He has guest edited Special Issues on Emerging Technologies for Data Security and Privacy in IEEE NETWORK and IEEE INTERNET OF THINGS JOURNAL.

Liehuang Zhu (Senior Member, IEEE) received the Ph.D. degree in computer science from the Beijing Institute of Technology, Beijing, China, in 2004. He is currently a Professor and the Dean of the School of Cyberspace Science and Technology, Beijing Institute of Technology. He has published more than 100 peer-reviewed journals or conference papers, including more than ten IEEE/ACM TRANSACTIONS papers. His research interests include security protocol analysis and design, wireless sensor networks, and cloud computing. He has been granted a number of IEEE best paper awards, including IWQoS 17' and TrustCom 18'.

Jiankun Hu (Senior Member, IEEE) is currently a Professor with the School of Engineering and IT, University of New South Wales, Canberra, Australia. He is an invited Expert with the Australia Attorney-General's Office, assisting with the draft of the Australia National Identity Management Policy. He has many publications in top venues, including IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, *Pattern Recognition*, and IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS. His research interests include the field of cybersecurity covering intrusion detection, sensor key management, and biometrics authentication. He has served on the Panel on Mathematics, Information, and Computing Sciences, Australian Research Council, Excellence in Research for Australia (ERA), and Evaluation Committee, in 2012. He is an Associate Editor of IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY.