Applied Intelligence
https://doi.org/10.1007/s10489-021-02453-9

LSTM-TC: Bitcoin coin mixing detection method with a high recall

Xiaowen Sun[1] · Tan Yang[1] · Bo Hu[1]

Accepted: 20 April 2021
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2021

Abstract
Coin mixing is a class of techniques used to enhance Bitcoin transaction privacy, and those well-performing coin mixing algorithms can effectively prevent most transaction analysis attacks. Based on this premise, to have a well-functioning transaction analysis algorithm requires coin mixing detection with a high recall to ensure accuracy. Most practical coin mixing algorithms do not change the Bitcoin protocol. Therefore, the transactions they generate are not fundamentally different from regular transactions. Existing coin mixing detection methods are commonly rule-based that can only identify coin mixing classes with well-defined patterns, which leads to an overall low recall rate. Multiple rules could improve the recall in this situation, yet they are ineffective for new classes and classes with ambiguous patterns. This paper considers coin mixing detection as a transaction classification problem and proposes an LSTM Transaction Tree Classifier (LSTM-TC) solution, which includes feature extraction and classification of Bitcoin transactions based on deep learning. We also build a dataset to validate our solution. Experiments show that our approach has better performance and the potential for discovering new classes of coin mixing transactions than rule-based approaches and graph neural network-based Bitcoin transaction classification algorithms.

Keywords Bitcoin · Coin mixing detection · CoinJoin · LSTM · Transaction tree · Bitcoin address clustering

1 Introduction

Bitcoin, the first blockchain implementation, is still highly sought after until today, and in early 2021 the price of Bitcoin is at a new high. At the same time, more applications based on blockchain technology are being developed [3, 29]. In this context, the security and privacy issues of blockchain should receive more attention. This paper focuses on the detection of coin mixing, which is essential in Bitcoin privacy enhancement.

All data in a public blockchain like Bitcoin is open to anyone, and the users' privacy depends entirely on anonymity, i.e., it is difficult to link data on the blockchain with real-world entities.

However, the structure of Bitcoin transactions makes it possible to infer address relationships from transaction data. Bitcoin transactions can have multiple inputs and outputs, and each output points to an address. Except for Coinbase transactions that generate new coins, all other transactions must reference unspent transaction outputs (UTXO) as inputs. Once the transaction is confirmed, the input coins are transferred. The destination addresses and the amounts will be specified in each output. Figure 1 shows the relationship between blocks, transactions, and addresses in Bitcoin.

All Bitcoin transactions can be viewed as a network [26], more specifically, a directed acyclic graph. Although there are studies to analyze its network topology to obtain address relationships, the more popular approach is to analyze individual transactions. The most common scenario is when a user does not have enough coins in a single UTXO and needs to use multiple UTXOs as one transaction's inputs. This action exposed that addresses corresponding to

The dataset is available here: https://github.com/sxwxs/BitcoinCoinMixingDataSetWithRuleBasedLabel

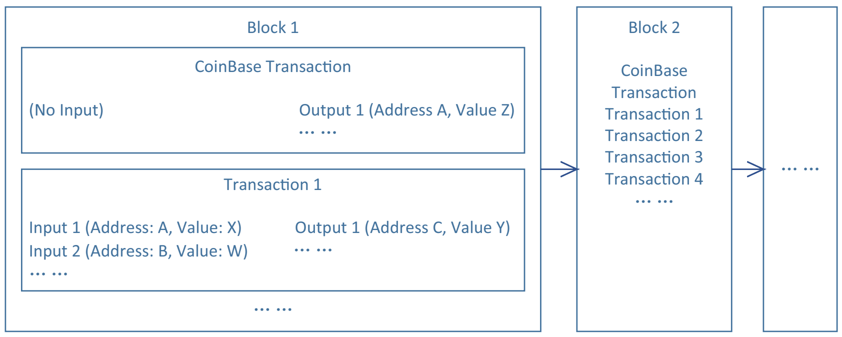
✉ Tan Yang
tyang@bupt.edu.cn

Xiaowen Sun
sxw@bupt.edu.cn

Bo Hu
hubo@bupt.edu.cn

[1] State Key Laboratory of Networking and Switching Technology, School of Computer Science (National Pilot Software Engineering School), Beijing University of Posts and Telecommunications, Beijing, 100876, China

Fig. 1 Bitcoin data structure



these UTXOs belong to the same person. This method of inferring multiple addresses relationship, which is known as the common-input-ownership heuristic, was proposed by Satoshi Nakamoto in the Bitcoin white paper [21]. This heuristic rule implies that if a transaction has multiple different addresses as input, then all those addresses must belong to the same user. For example, in general, we can assume that address A and address B in block 1, transaction 1 in Fig. 1 belong to the same user. When multiple addresses are found to belong to the same user, the set of these addresses is called a wallet. The reason it works is before broadcasting, all transactions must be signed using the private keys of all the input addresses, and only the person who owns those addresses can complete the signature. It is also one of the most widely used and effective methods for address clustering and address relationship determining.

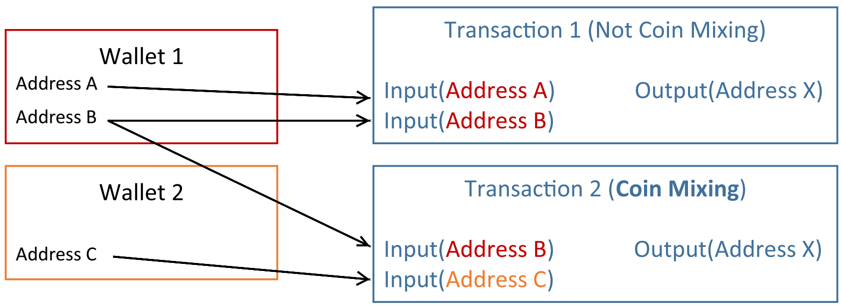
For against individual transactions analysis, the coin mixing technique is one of the suitable counterattacks. In 2013 Maxwell G. proposed an approach that allowed different users to negotiate and sign the same transaction [16]. The approach is known as a CoinJoin, the first widely used coin mixing algorithm. The way it works is either merging multiple users' coins and splits them into multiple shares, then returning to the original owners, or combining multiple transactions that involving multiple users into a single transaction. As shown in Fig. 2, if the addresses of different wallets are used as input for the same transaction, then the transaction is a coin mixing transaction. Coin mixing transactions have no distinctive features, and the relationship between Bitcoin addresses is difficult to obtain accurately, so it is difficult to identify coin mixing transactions in general. CoinJoin transaction not only enhances Bitcoin transaction privacy but also reduces the fees and number of transactions. With such advantages, CoinJoin and other coin mixing methods are gradually gaining traction.

Nowadays, coin mixing transactions are so common that almost every Bitcoin addresses clustering method requires a coin mixing detection algorithm. If the common-input-ownership heuristic is enforced indiscriminately for all multi-input transactions, it can lead to many address clustering errors. Not using this heuristic, in turn, can result in the loss of a large number of relationships, leading to unsatisfactory results. Address clustering requires a high recall coin mixing detection algorithm to reduce incorrect connection of addresses, and according to the results in Section 5.3, the coin mixing detection algorithm has a significant impact on the address clustering result.

Recently, Weber M. et al. proposed a GCN-based Bitcoin transaction classifier to detect illegal transactions [35]. Their method already has a high performance, both on their proposed Elliptic dataset and on the coin mixing dataset presented in our paper. However, to apply a method to Bitcoin address clustering, some additional requirements need to be met.

1. The speed of running this method and preprocessing of data cannot be slower than the speed of new Bitcoin blocks being generated.
2. A high recall is required to ensure accurate results.

Fig. 2 Coin mixing transaction and normal transaction



3. The methods need to have the potential to discover new classes of coin mixing to adapt to the evolution of coin mixing methods

This paper aims to propose a supervised learning based coin mixing detection algorithm with a high recall rate for identifying coin mixing transactions in Bitcoin and to help analyze Bitcoin address relationships.

We have three motivations to do this work.

1. Coin mixing technology is constantly evolving, but due to the tamper-proof nature of the blockchain, records of coin mixing transactions generated using obsolete technology cannot be removed. There are already many ways to identify these obsolete coin mixing transactions. So, we can get positive cases using existing methods. Figure 3 shows that the once-active SharedCoin transactions (one kind of coin mixing transaction) virtually disappeared after some time in 2016, as it has proven to be vulnerable.
2. Rule-based coin mixing detection methods can be easily bypassed and need to be updated manually to cover new classes.
3. We believe that even different classes of coin mixing transactions should have some of the same patterns. Especially from the point of view of transaction sequences, because a coin mixing transaction can hide in various ways, but its purpose is stationary, i.e., mixing coins from different users.

We view the coin mixing detection as a Bitcoin transaction classification problem, and propose a transaction tracing algorithm where a transaction is selected to start with, and two trees consisting of the precursor and successor transactions of this transaction can be obtained. Then the trees are converted into sequences using an aggregate function. Finally, the features of the sequence will be feed into LSTM to get the label. Experiments show that our model can process Bitcoin transaction data in real-time and has the potential to identify new coin mixing classes.

2 Related work

2.1 Bitcoin privacy-enhancing techniques

Bitcoin's privacy-enhancing technology includes coin mixing, is always evolving. The Bitcoin wiki gives several methods to improve Bitcoin privacy [1], two of which are closely related to our work and are also widely used.

1. Bitcoin owners are advised not to reuse addresses, meaning that each address should only be used twice, once to receive coins and another to send all those coins.
2. Bitcoin recommends that users endeavor to use CoinJoin for transactions, which not only saves on fees to some extent but also prevents adversaries from identifying address relationships and tracking where coins go.

The first suggestion would result in a large number of one-off addresses, which contains only a small amount of information, and the number of addresses in the system would increase continuously. The other impact is

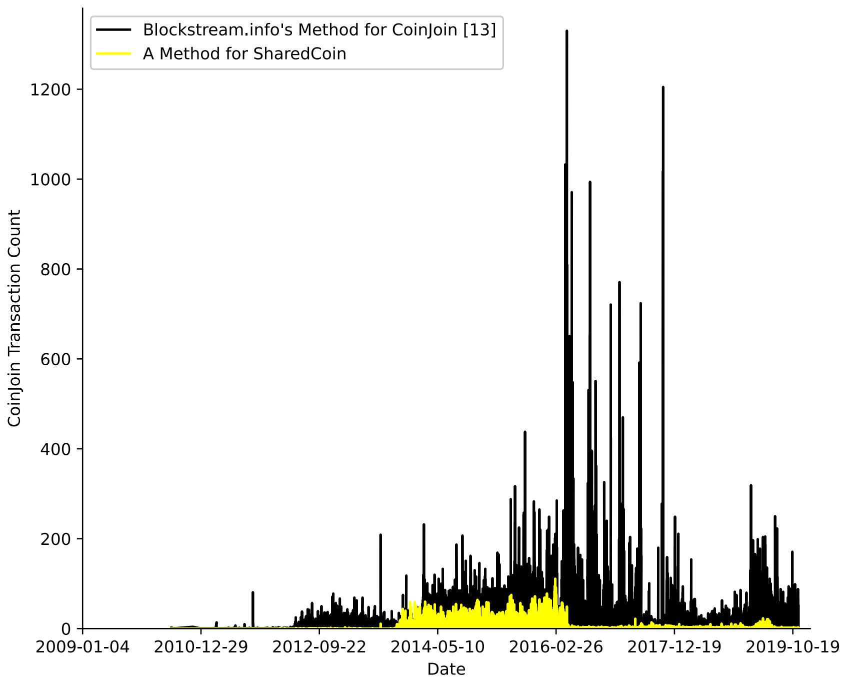


Fig. 3 Coin mixing transaction count in bitcoin blockchain

subsequent of many transactions will construct a tree, or even a chain of transactions rather than a network, as each address has one transfer in and one transfer out. Based on this feature, we propose a tree-based feature extraction algorithm. Comparing with methods such as graph embedding, our algorithm run faster, i.e., linearly with the number of transactions, and is more suitable for long transaction chains.

The second proposal leads to a much more significant impact related to CoinJoin, one of the most common coin mixing methods, also the key point of this paper. The common-input-ownership heuristic is 100% accurate when coin mixing transactions are not considered. However, when dealing with coin mixing transactions, the common-input-ownership heuristic can make serious errors, i.e., associating addresses that have no connections with each other. Current address analysis algorithms are almost impossible to work with if coin mixing transactions cannot be effectively identified.

2.2 CoinJoin

CoinJoin was first introduced by Maxwell G. in bitcointalk.org in August 2013 [16]. A few months earlier, he had described a similar approach in another post [17]. Creating a regular Bitcoin transaction requires the user to sign the transaction with the private keys of all inputs and then broadcast it. When creating a CoinJoin transaction, however, multiple users are allowed to agree on the content of the transaction and then sign the transaction separately, because no one will sign a transaction that they don't approve of, and only if all parties to the transaction sign will the transaction be valid. CoinJoin is relatively easy to implement. There are already several implementations of CoinJoin, such as CoinWitness [18], and some organizations offer CoinJoin services, like SharedCoin [25]. So CoinJoin is very widely used [30]. CoinJoin does not expand the Bitcoin protocol, and to the outside world, CoinJoin transactions do not look any different from normal Bitcoin transactions. Since the Bitcoin network is completely distributed, it is also impossible to detect CoinJoin by the network information. However, coin mixing transactions generated by some naive CoinJoin algorithms can be identified by matching the input and output values.

The CoinJoin algorithm is continuously improving. In 2017, Maurer F. K. et al. proposed a method of splitting outputs to prevent adversities from link CoinJoin transactions' inputs and outputs based on knapsack mixing [15]. It is difficult to identify transactions generated by a well-implemented CoinJoin algorithm based on information about a single transaction alone.

2.3 CoinJoin detection

In 2013 Blockchain.info started offering the SharedCoin service and published their code. However, in 2014, Atlas K. pointed out that SharedCoin was vulnerable [5]. Blockchain.info later stopped the service and removed the code [31]. Atlas K. has created the tool named CoinJoin Sudoku for identifying and deciphering SharedCoin transactions and has published the code [4, 6]. However, we were unable to reproduce his work because some of the data interfaces he used were no longer valid. CoinJoin Sudoku relies on data not just from the Bitcoin blockchain, but also some additional data provided by BlockChin.info's API, such as relayedByIP and type, but now the API seems to return only constant values, we looked up a lot of data, but the relayedByIP is always "127.0.0.1" and the type is all "0".

Blockstream.info using a rule to detect CoinJoin transactions [13].

First, the CoinJoin transaction must have more than one input. Then a $Target$ value for the transaction should be calculated. It is half the number of outputs capped between 2 and 5. If the number of occurrences of the plurality of the output value is greater than or equals to $Target$, the transaction will be view as CoinJoin like transaction. Algorithm 1 is a pseudo-code description of this method.

Algorithm 1 CoinJoin like transaction detection.

```
Input: Target Transaction;
Output: If the transaction is CoinJoin like;
1:  if $Transaction.Inputs.length < 2$ then return False
2:  else
3:      Target = Min(Max(Transaction.Outputs.length / 2,
        2), 5)
4:      Counter = HashMap()
5:      for each $output \in Transaction.Outputs$ do
6:          if $output.ValuenotinCounter$ then
7:              Counter[output.Value] = 1
8:          else
9:              Counter[output.Value] += 1
10:         end if
11:         if $Counter[output.Value] >= Traget$ then
12:             Return True
13:         end if
14:     end for
15: end ifreturn False
```

Blockstream.info's approach uses only information from a single transaction, runs fast, and has high precision. However, the problem is that it can only catch this one class of CoinJoin, and it can completely bypass this detection by avoiding producing the same value in output when splitting the coins.

Meiklejohn S. et al. used a fuzzy method to identify CoinJoin transactions, i.e., transactions with more than 5 inputs and more than 5 outputs are considered coin mixing transactions [19]. This method identifies CoinJoin based on single transaction, but it is obvious that this method suffers from a precision problem, as many normal Bitcoin transactions will also have a higher number of inputs and outputs.

Maksutov A. A. et al. proposed a method based on transaction graph and user graph in 2019 [14]. Firstly, they build a transaction graph and user graph. The definition of their transactions graph is the same as shown in Fig. 5. Their user graph describes the interaction between users over time. The initial user graph is actually an addresses graph. After that, addresses can be merged according to some rules. The paper then theoretically demonstrates the possibility of finding input-output mappings of actual sub-transactions from coin mixing transactions.

2.4 Other coin mixing method

There are many other classes of coin mixing methods [8–10, 28, 34, 38, 39]. Mixcoin is a centralized method of coin mixing [9]. It requires a dedicated server to perform the mixing, which introduces a lot of risks. Blindcoin is an improvement to MixCoin that reduces the risk of user privacy leaks from the mixer server [34]. Bissias et al. proposed a method resisting Sybil attack named Xim [8]. Ziegeldorf J. H. et al. proposed a decentralized coin mixing method that can be applied to a variety of cryptocurrencies. CoinShuffle is based on CoinJoin, and it improves privacy within coin mixing participants [28]. CoinParty using multiple one-to-one transactions replacing the group transactions used in CoinJoin, CoinShuffle, etc. [38]. ZoreJoin combines ZoreCoin and CoinJoin to overcome some of their disadvantages [10].

Although the implementations vary widely and the characteristics of the generated transactions are different, we treat multiple coin mixing as a single class in the transaction classification problem. We expect the model to extract universal features of coin mixing transactions and achieve a high recall of overall coin mixing transactions.

2.5 Bitcoin address clustering

There has been a lot of work on address clustering [2, 11, 12, 20, 22, 26, 27, 32, 37]. All of these studies mentioned or used the common-input-ownership heuristic. Similarly, if the heuristic is used, some sort of coin-mixing detection method has to be used. We have tried using all multi-input transactions up to 510 thousand blocks and got an extremely large address set with more than 12 million addresses which are related to coin mixing.

Zheng B. et al. simply discards all transactions that have more than one output when applying the common-input-ownership heuristic [37].

2.6 Bitcoin transaction classification

We consider coin mixing transaction detection as a transaction classification problem. We found one study on Bitcoin transaction classification. Weber M. et al. proposed a method based on graph convolutional networks (GCN) and random forests to perform binary-classify for Bitcoin transactions [35]. They also provide a dataset named Elliptic dataset includes 200 thousand Bitcoin transactions. The labels are either legal or illegal and come from a company's internal data (most transactions are unlabeled).

Their paper presents the concept of Local Features (LF), which are the characteristics of individual transactions, such as the fee, address count, value of inputs, and outputs of the transaction. This concept will be used in the following.

2.7 Transaction and address classification other than bitcoin

There has been some recent research using machine learning to classify Ethereum transactions, and wallets [7, 36]. Wu J. et al. and Baek H. et al. both crawl public labeled data from some websites to build their dataset. Wu J. et al. using a new network embedding algorithm they proposed to get the features of transactions and then using the one-class support vector machine to classify the transactions. Baek H. et al. using the random forest to perform anomaly detection.

3 Methodology

We found a total of 132,480 coin mixing transactions between Block 270,000 (2013-11-16 22:09:06 UTC) and Block 300,000 (2014-05-10 06:32:34 UTC) using two existing rules i.e., CoinJoin transaction detection rule from blockstream.info and a SharedCoin detection rule (shown in Fig. 3).

These two rules are thought to have performed well in early Bitcoin transactions. This was mainly because at that time, there was less research on coin mixing algorithms, users did not have much choice, and the algorithms were simple back then with significant patterns that could be discovered with rules.

But SharedCoin transactions faded away after 2016 due to security concerns. It was replaced by more complex coin mixing algorithms. We hope our model can learn the features of coin mixing transactions from data prior to 2016 and can work on data after 2016.

From Fig. 3, it can be noted that the rule for SharedCoin has barely recalled any coin mixing transactions after some time in 2016, as users have migrated to coin mixing classes that are more difficult to detect. Rule-based detection algorithms require constant manual updates to avoid this problem. Another problem is that the difference between transactions generated by the coin mixing algorithm and those generated by normal users is becoming less and less obvious. There are even coin mixing algorithms that generate multiple transactions at a time, which makes the design of rule-based methods even more difficult. But our model can handle these problems very well because the input of our model is two transaction trees containing a large number of transactions around the target transaction.

Strictly speaking, the information of a single transaction itself, i.e., the Local Features mentioned in Section 2.6, is not enough to determine coin mixing. Suppose there are two transactions, A and B, except that the input addresses are identical, but all the input addresses of transaction A belong to the same entity, but the input addresses of transaction B belong to multiple entities, then transaction A is a coin mixing transaction, but B is not, although their Local Features are identical.

Our approach will focus on the preceding transaction tree, and the succeeding transaction tree of the target transaction, which may involve tens of millions of transactions, with a tree depth or sequence length of up to tens of levels, and the model can have a grasp enough range of information.

In contrast, the rule-based approach can only utilize information about a single or a few transactions, so although the dataset we use to train the model is built using rule-based methods, the model is concerned with information over long distances while rules are concentrated on Local Features of the target transaction. We want the model to get enough information from train data and apply it to the identification of new classes of coin mixing.

4 LSTM transaction-tree classifier

Our solution consists of three parts: transaction tree extractor, transaction tree serializer, LSTM classifier, as shown in Fig. 4.

The input of the algorithm is a transaction, and the output is whether the transaction is coin mixing or not. The algorithm works as follows.

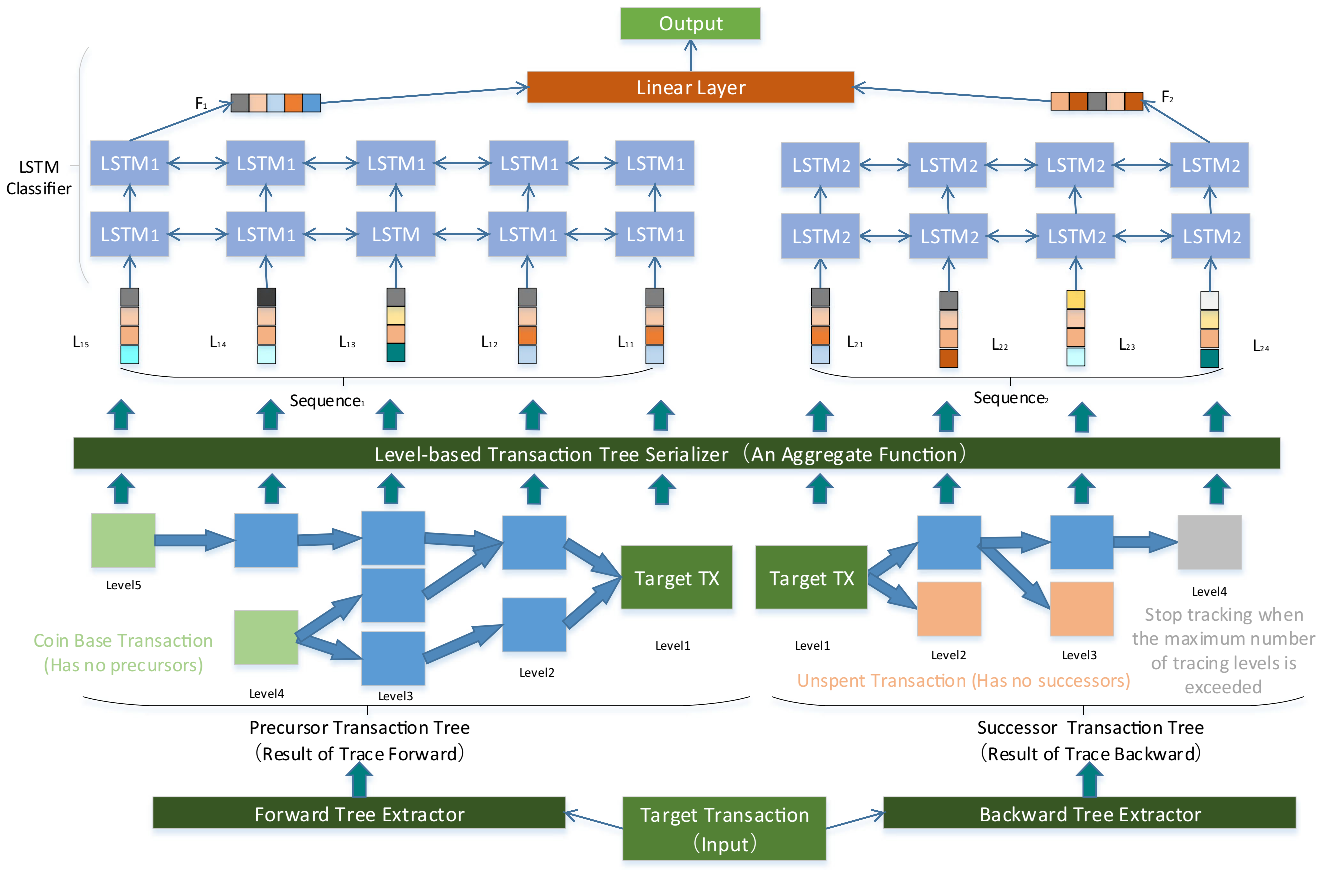

Output
Linear Layer
F₁
F₂
LSTM Classifier
LSTM1
LSTM
LSTM2
L₁₅
L₁₄
L₁₃
L₁₂
L₁₁
L₂₁
L₂₂
L₂₃
L₂₄
Sequence₁
Sequence₂
Level-based Transaction Tree Serializer（An Aggregate Function）
Level5
Level4
Level3
Level2
Level1
Target TX
Coin Base Transaction (Has no precursors)
Unspent Transaction (Has no successors)
Stop tracking when the maximum number of tracing levels is exceeded
Precursor Transaction Tree (Result of Trace Forward)
Successor Transaction Tree (Result of Trace Backward)
Forward Tree Extractor
Target Transaction (Input)
Backward Tree Extractor

Fig. 4 Overview of our solution

1. Extracting the transaction tree from the Bitcoin blockchain data, based on the input transactions (Target Transaction), at the N level before (Trace Forward, Precursor Transaction Tree) and M level after (Trace Backward, Successor Transaction Tree) the transaction.
2. Each level of transaction trees is converted into a fixed-length vector using an aggregation function, and the aggregated generated features are manually specified. The outputs of this step are two sequences ($Sequence_1$ for precursor tree, $Sequence_2$ for successor tree).
3. The two sequences from the second step are fed into the two LSTMs, resulting in two fixed-length vectors, and then pass the two vectors to a linear layer to obtain the result.

We define the input transaction as Target Transaction, the maximum number of layers of the precursor transaction tree used in this algorithm to be N and the maximum number of layers of the successor transaction tree to be M, the two sequences obtained by serializing the transaction trees are:

$$Sequence_1 = [L_{11}, ..., L_{1n}](n <= N) \tag{1}$$

$$Sequence_2 = [L_{21}, ..., L_{2m}](m <= M) \tag{2}$$

$Sequence_1$is the result of the precursor transaction tree (trace forward), and $Sequence_2$ is the result of the successor transaction tree (trace backward). L_{11} and L_{21} correspond to the first level of the two transaction trees, which contain only one transaction, i.e., the Target Transaction.

The two LSTM models are defined as $LSTM_1$ and $LSTM_2$, and the Linear Layer is denoted as $Linear$. Then our model can be expressed as:

$$F_1 = LSTM_1(Sequence_1) \tag{3}$$

$$F_2 = LSTM_1(Sequence_2) \tag{4}$$

$$Output = Linear(F_1, F_2) \tag{5}$$

where [.; .] denotes the concatenation of two vectors.

It should be noted that N and M are not necessarily the same here, and different N and M were set in our experiments. A large M not only leads to expensive computations for early transactions, but a bigger problem is that it limits the use of the algorithm to newly generated transactions. Because newly generated transactions need time to be referenced by successor transactions. If a large M is specified, a long time may have to be waited before there are enough subsequent transactions to run the algorithm with confidence.

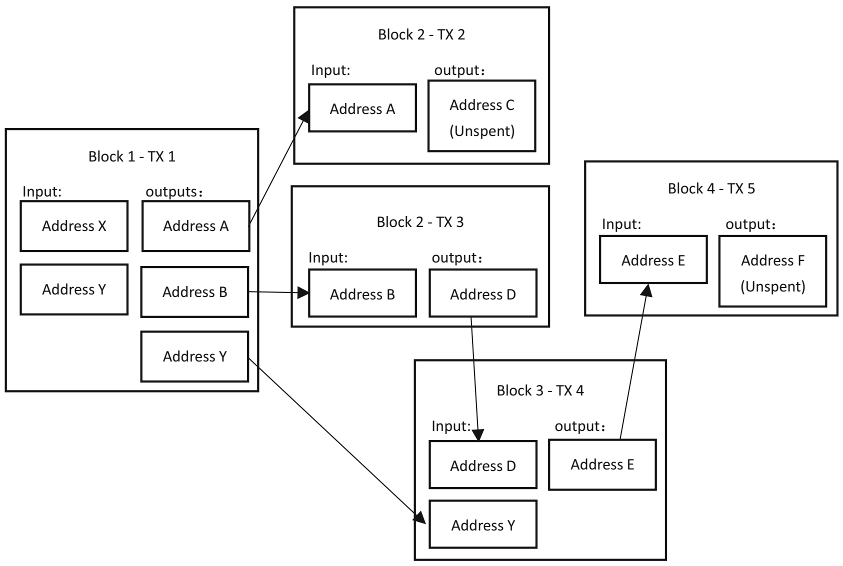
4.1 Transaction tree extractor

We proposed a new method to represent the transaction directed acyclic graph (DAG) as a tree, by allowing duplicated transactions to appear in different parts of the tree. Figure 5 shows a Bitcoin transaction DAG, and the method of Maksutov A. A. et al. view Bitcoin transactions as a DAG.

In order to convert the DAG to a tree graph, if there is a transaction that is the successor of more than one previous transaction, for example, TX 4 in Fig. 5, it will be drawn multiple times, but only the first one can have subtrees, i.e., the subtree of other duplicated nodes will be omitted. Figure 6 shows the transaction tree arranged by our method.

Performing tracing on a large number of transactions at the same time may consume a lot of memory. We extract the precursor and successor relationships of TXIDs from the raw blockchain data, and then load them in chunks and iterate through them to build the transaction tree.

Fig. 5 Transaction DAG



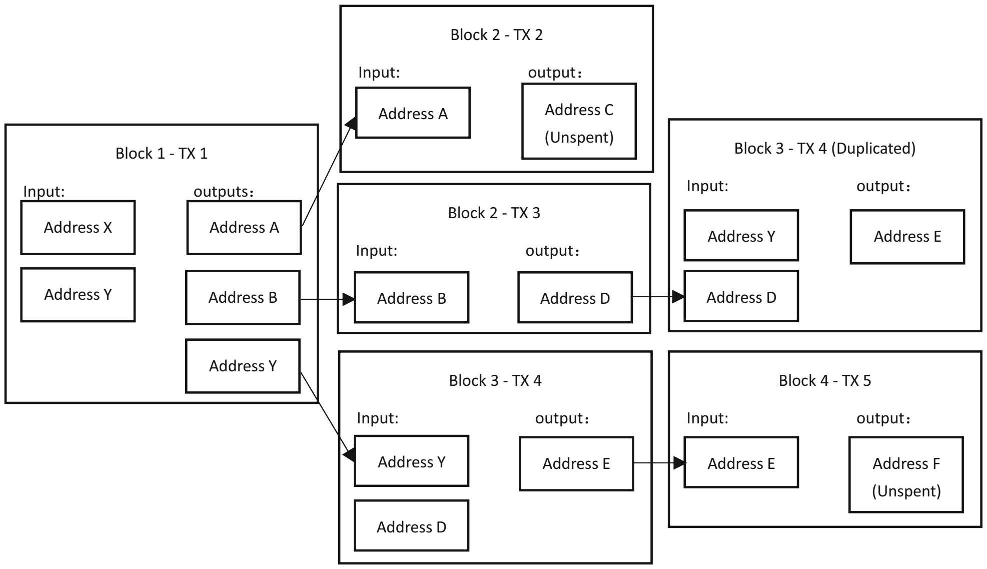


Fig. 6 Transaction tree

The transaction tree will be calculated and saved in the order of the levels, and the first level only contains the target transaction.

$$TreeLevel_1 = TransactionSet([TargetTransaction]) \tag{6}$$

$$TreeLevel_i = TransactionSet(TreeLevel_{i-1}) \tag{7}$$

4.2 Transaction tree serializer

Each layer of the transaction tree was summarized into a fixed-length vector using a manually designed aggregation function, which results in a sequence of length equal to the number of tree levels.

Each transaction tree may involve tens of millions of transactions, and in extreme cases, one layer may include more than 10 million transactions. This is a computational challenge. To save computational time consumption, the 15 features (such as the sum of input amounts, the sum of output amounts, processing fees, number of input addresses, number of output addresses, number of inputs, number of outputs, etc.) of a single transaction will be completed in the preprocessing stage, i.e., all the transactions that may appear in the transaction tree will be preprocessed, and the index will be constructed in memory.

Then we summarized 78 aggregation features of each level of the transaction tree, including transaction count (the number of transactions, the number of Coinbase transactions, and the number of non-Coinbase transactions), and statistical values (total sum, mean, standard deviation, maximum value, minimum value) for each 15 features of transactions in this level. A memory-based multi-threaded program was implemented using C++ to calculate them fast. Figure 7 illustrates how the transaction tree feature extractor works.

$$Sequence_i = AggregateFunction(TreeLevel_i) \tag{8}$$

4.3 LSTM classifier

Two separate multilayer LSTMs are used to process sequences from two transaction trees and generate two fixed-length vectors. The LSTM model is trained by connecting the output vectors to a linear layer convert the two vectors to a single score represent if the target transaction is coin mixing. The structure of our LSTM classifier model is shown in Fig. 8.

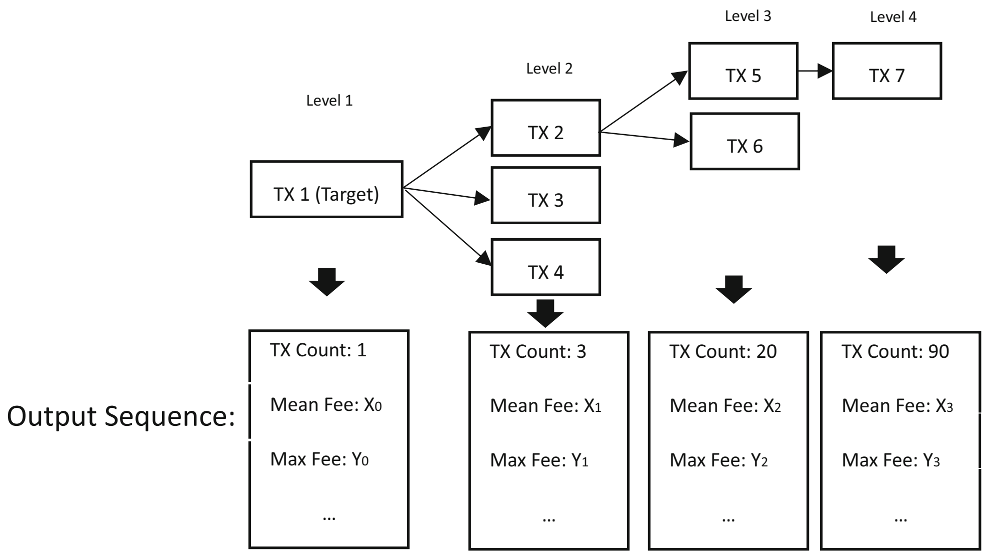
We believe that using LSTM models to extract transaction features is advantageous for graph embedding algorithms that have been previously applied to transaction classification, even if the Bitcoin transaction relationship is a network or more specifically a directed acyclic graph.

First, the size of the Bitcoin transaction graph is very large, and it is difficult to handle such a large graph and update the graph embedding features in real-time.

Second, and more importantly, LSTM can capture some long-range characteristics. We believe that coin mixing transactions are essentially the process of converging and separating coins of different origins and that there are certain types of coin mixing that are not completed in a single transaction, so these characteristics may be better captured using LSTM.

Third, we believe that particular transactions in the Bitcoin system, such as coin mixing transactions and certain transactions inside exchanges are strongly characterized,

Fig. 7 Transaction tree serializer



and that their characteristics, such as the number of inputs and outputs, differ greatly from most other common transactions. The number and amount of inputs and outputs of these transactions may differ significantly from those of ordinary transactions by more than a few orders of magnitude, and considering them all together may lead to the blurring of the characteristics of a large number of ordinary transactions. For example, some large transactions of some exchanges can have thousands of input and output but a typical Bitcoin transaction often has only a few inputs and outputs. In our approach, the problem can be succinctly avoided by excluding certain categories of transactions based on rules or a label database at the time of transaction tree extraction, which will be the next step in our research.

Finaly, LSTM Classier uses sequence features, and the sequence features are obtained from transaction trees, the range of data accessible to the model can be controlled by simple choosing of tree levels, as we compare the results of using the all tree levels and deleting the first level of the tree in our experiments.

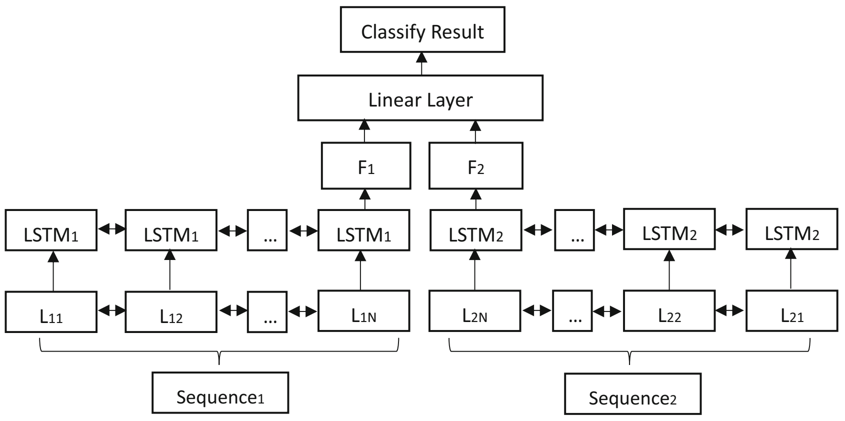
5 Evaluation

5.1 Dataset

We created two datasets. The first one is much earlier in time and contains a large number of easily detectable coin mixing transactions, which we labeled using rules and then used to train the model. The second dataset is unlabeled and much later in time. We use this dataset to test the effectiveness of our model and some other methods of coin mixing detection. We will also perform common-input-ownership heuristic based address clustering on the second dataset to demonstrate the benefits of our coin mixing detection algorithm.

The first dataset is the training set that contains 264,960 transactions between block 270,000 (2013-11-16 22:09:06 UTC) and block 300,000 (2014-05-10 06:32:34 UTC), labeled by Blockstream.info's method of CoinJoin detection and a method of SharedCoin detection. In this period, SharedCoin is active.

Fig. 8 LSTM classifier



The second dataset is the testing set which contains 168,192 transactions between 411,500 (2016-05-12 23:11:14 UTC) and Block 412,200 (2016-05-17 19:50:59 UTC). The second period contains 700 blocks, which is the time when SharedCoin was inactive.

On both datasets, Coinbase transactions and transactions with only a single input are excluded, as they are impossible to be coin mixing transactions and are also easy to filter by rules.

For the training set, we first label coin mixing transactions as positive cases using previously mentioned rules. But the percentage of coin mixing in all transactions is very low, only about 1.18%. To ensure the balance of positive and negative samples, negative samples are sampled in the remaining transactions according to a rule. First, the 17-level precursor transactions of the positive sample are removed (shown in Fig. 9), which also ensures that the remaining transactions are less likely to be coin mixing transactions, since the rules we use do not guarantee 100% recall for coin mixing transactions.

The dataset also contains a large number of additional transactions associated with these transactions described above, so the actual number of transactions is 116,537,811. These transactions are preprocessed to get features represented using normalized numerical values, stripped of the real TXID, and labeled with integer IDs. For each transaction there are 63 local features provided. In the following experiments, both the graph neural network approach as well as the LSTM-TC use only the above-mentioned transactions and these 63 features.

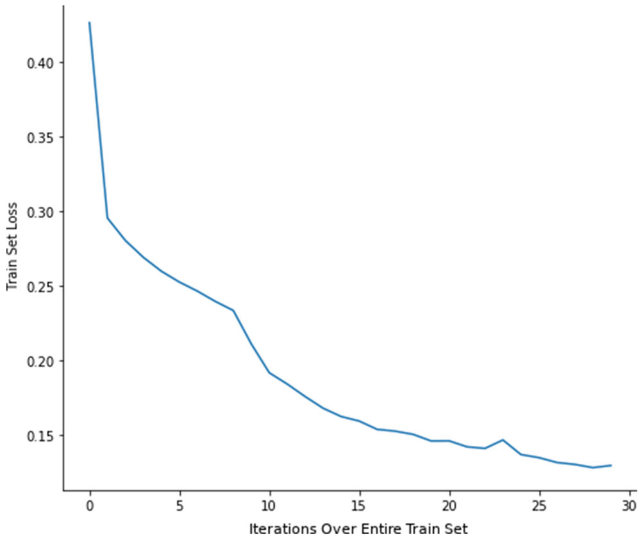


Fig. 10 The variation of loss

5.2 Experiment details

In this section, we will introduce the details and hyperparameters in our experiment. In the Transaction Tree Extractor part, we trace forward (to earlier time) from target for up to 30 levels (N=30) and trace back up to 10 levels (M=10).

Fig. 9 Remove 17-levels precursor transactions of positive cases

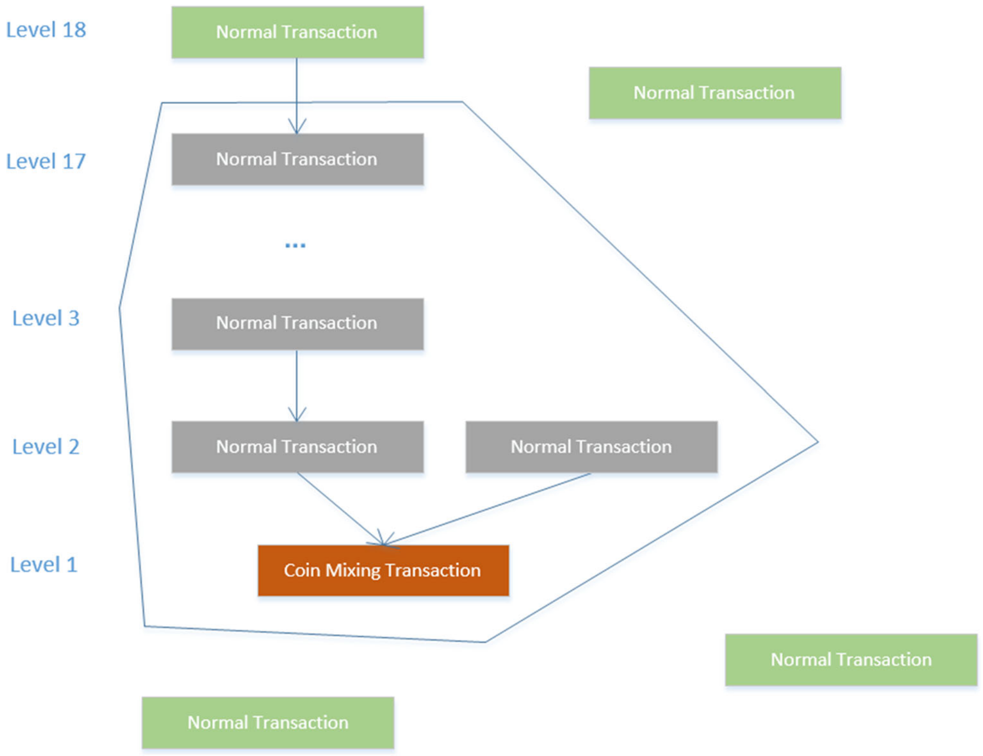


Table 1 The result of address clustering using different coin mixing detection method

Coin Mixing Detection Method	Recall Ratio	Available TX for clustering	Wallet Count
Method of Zheng B. et al. [37]	84.89%	25,411	481,696
LSTM-TC (threshold 0.01) + Blockstrea.info's Method of CoinJoin +A Method for SharedCoin Detection	51.87%	80,956	344,052
LSTM-TC (threshold 0.01)	50.89%	82,592	337,387
Random Forest Classifier using GCN Feature [35]	14.81%	143291	168,568
Blockstrea.info's Method of CoinJoin Detection [13]	2.12%	164,633	102,300
A Method for SharedCoin Detection	0.02%	168,161	88,798
No Coin Mixing Detection	0%	168,192	88,498

Bidirectional LSTM model with 3 layers provided by PyTorch [23] was used. The hidden layer dimension of the LSTM model is 128. The optimizer is SGD. The learning rate is 0.53. The metrics for evaluation are F1-score, precision, recall, and accuracy, which are implemented by Scikit-learn [24].

Model is training on a GPU of NVIDIA GeForce RTX 2080 Ti for 30 epochs. The variation of loss is shown in Fig. 10.

5.3 Address clustering using our model

We perform common-input-ownership address clustering experiments based on LSTM-TC, Random Forest Transaction Classifier using GCN Feature [35] (both of them are trained on the training set only) and two rule-based coin mixing detection method on the testing set. We also tried methods from other studies on Bitcoin address clustering. We implemented a fast address merging algorithm using the disjoint-set. The recall ratio (means the percentage of transactions considered as coin mixing to all transactions), available transactions for clustering, and wallet count generated by clustering is shown in Table 1.

When using the common-input-ownership, coin mixing transactions need to be excluded. So, the more coin mixing transactions detected by the coin mixing detection algorithm, the fewer transactions will be available for clustering.

A wallet here refers to a collection of addresses that are under the control of the same entity. The minimum number of wallets can be obtained when not using coin mixing detection. This indicates that many wallets are incorrectly merged due to the misleading of coin mixing. And the two rule-based coin mixing detection algorithms have high precision but miss more coin mixing classes and get results containing errors. And just the opposite, the method of Zheng B. et al. gets the highest number of wallets. Because this method excludes all transactions with multiple outputs, which is an extremely strict condition that leads to a large amount of valuable information being lost.

Our approach is recalling transactions suspected to be coin mixing and leaving a large number of normal

Table 2 Fitting result on training data

Model	Features Used	Threshold	Precision	Recall	F1
LSTM-TC	Forward + Backward	0.5	0.966	0.961	0.964
		0.01	0.826	0.989	0.900
	Forward Only	0.5	0.871	0.920	0.895
	Forward+Backward (First level removed)	0.5	0.982	0.953	0.967
		0.01	0.826	0.989	0.900
	Forward Only (First level removed)	0.5	0.860	0.921	0.890
Random Forest Classifier using LINE Feature	Forward + Backward	-	0.579	0.654	0.614
	Forward Only	-	0.523	0.153	0.236
Random Forest Classifier using GCN Feature[35]	-	-	0.976	0.990	0.983

transactions for address clustering. Furthermore, it is possible to control the number of recalls by modifying the threshold.

Both our model and method of Zheng B. et al. have significantly higher recall ratios than the rule-based methods. We compare the transactions recalled by these two over the transactions recalled by the rule. 30 transactions were sampled from each of these two sets and were checked manually. After manually checking 47% of the transactions in the results of LSTM-TC are likely to be coin mixing transactions, while the result of Zheng B. et al.'s method is 33%. So our method can better balance precision and recall to help address clustering achieve better results.

5.4 Fitting on the training data

We compared the performance of our model with Random Forest Classifier using GCN Feature. We also try using LINE (Large-scale information network embedding) to replace GCN [33]. The results are shown in Table 2.

The method of Weber M. et al. achieves good performance on the Elliptic Data Set which is a Bitcoin illicit transaction dataset.

According to the results of the paper of Weber M. et al., Local Features of transactions, i.e., the number of inputs and outputs, amounts, fees, etc., can greatly improve the performance of transaction classification. But our training set's labels are obtained using Local Features (rule-based coin mixing detection relies entirely on local features). So, we do not want our model to get the Local Features of the target transactions. The transaction tree feature extractor already fuzzes the Local Features. But we are concerned that the first level of the transaction tree will leak these Local Features, because the first level of the transaction tree contains only one transaction, the target transaction. So, we also try to use features with the first level of the transaction tree removed.

The results show that removing the first level of transaction trees does not affect the performance of LSTM-TC, indicating that the model determines the coin mixing from sequence features rather than Local Features.

The Random Forest Classifier using GCN Feature can fit the training set well, but this is not necessarily the best result. Because on the test set, the coin mixing classes can be different. The problem of the GCN method may be over-fitting the characteristics of the coin mixing classes labeled by the rules, thereby missing more other classes of transactions. But recall is more important in this scenario. So, our model may perform better on the testing set.

Since the number of transactions is very large and LINE algorithms consume a lot of memory, we have modified the code from the origin LINE repository[1] to make it more memory-efficient without affecting performance. An open-source implementation[2] of GCN Bitcoin transaction embedding for the paper of M. Weber et al. is used.

Table 3 Memory consumption comparison of transaction tree feature extractor and LINE

Method	Features used	Tx count	Feature dim	Memory usage
Transaction tree	Backward	38 million	78	10 GB
Feature extractor	Forward	78 million		20 GB
LINE	Backward	38 million	200	66 GB
	Forward	78 million	(default)	130 GB

Our feature extraction algorithm is more memory-efficient than the LINE algorithm. Although in our approach, the features of all nodes need to be loaded into memory as well, only the features of the target transactions need to be generated, and the target transactions represent only a small fraction of all transactions, whereas the LINE has to generate and iteratively update the feature vectors of all transactions (even if the features of the vast majority of transactions will not be used by the classifier) (Table 3).

As for GCN graph embedding, its memory consumption is highly dependent on the range of nodes selected and requires GPU-accelerated computation, which we do not compare here.

6 Conclusion and future work

In this paper, we build a coin mixing dataset for Bitcoin. And we propose a new solution for Bitcoin transaction classification named LSTM Transaction Classifier (LSTM-TC), including related transaction extraction, serialization, feature extraction, and LSTM classifier. In terms of recall, LSTM-TC outperforms the rule-based coin mixing detection method, and Random Forest Classier using GCN Feature and LSTM-TC is more accurate than some arbitrary rules. LSTM-TC also outperforms the rule-based approach for detecting coins mixing in the address clustering experiments performed on the second dataset. Finally, our model has the potential to improve efficiency and algorithmic effectiveness by improving the transaction tree extraction algorithm.

Although we manually check a small amount of sampled data in the evaluation, there is an overall lack of human labeled coin mixing data in the experiments, and all training and evaluation data are labeled by coin mixing

[1] https://github.com/tangjianpku/LINE

[2] https://github.com/JungWoo-Chae/GCN_Elliptic_dataset

detection rules. Because, in many cases, it is not easy to distinguish between coin mixing transactions and normal transactions even by a human. Usually, for well-designed coin mixing, only the participants and those who have internal information can give a clear judgment. If more relevant data becomes available in the future, we would like to examine how our algorithm performs in diverse cases.

For some transactions, as mentioned in Section 4.2, some layers of the transaction tree may contain up to tens of millions of transactions, and we are concerned that the fixed-length features obtained by level-based transaction tree serializer cannot completely hold the features of such a large number of transactions. To alleviate the problem, in addition to improving the serialization algorithm, the transaction filtering idea we mentioned in Section 4.3 is also a possible option.

Funding This work is supported by the National Key Research and Development Program of China under Grant No.2020YFE0200500

Availability of data and material The data mentioned in the article are available here: https://github.com/sxwxs/BitcoinCoinMixingDataSetWithRuleBasedLabel

Declarations

Conflict of Interests The authors declare no conflict of interest.

References

1. Bitcoin wiki (2010) Privacy. https://en.bitcoin.it/wiki/Privacy
2. Androulaki E, Karame GO, Roeschlin M, Scherer T, Capkun S (2013) Evaluating user privacy in bitcoin. In: International conference on financial cryptography and data security. Springer, pp 34–51
3. Arjun R, Suprabha K (2020) Innovation and challenges of blockchain in banking: a scientometric view International Journal of Interactive Multimedia & Artificial Intelligence 6(3)
4. Atlas K (2014) Coinjoin sudoku repository. https://github.com/kristovatlas/coinjoin-sudoku
5. Atlas K (2014) Weak privacy guarantees for sharedcoin mixing service
6. Atlas K (2015) Coinjoin sudoku http://www.coinjoinsudoku.com
7. Baek H, Oh J, Kim CY, Lee K (2019) A model for detecting cryptocurrency transactions with discernible purpose. In: 2019 Eleventh international conference on ubiquitous and future networks (ICUFN). IEEE, pp 713–717
8. Bissias G, Ozisik AP, Levine BN, Liberatore M (2014) Sybil-resistant mixing for bitcoin. In: Proceedings of the 13th workshop on privacy in the electronic society, pp 149–158
9. Bonneau J, Narayanan A, Miller A, Clark J, Kroll JA, Felten EW (2014) Mixcoin: Anonymity for bitcoin with accountable mixes. In: International conference on financial cryptography and data security. Springer, pp 486–504
10. Chepurnoy A, Saxena A (2020) Zerojoin: Combining zerocoin and coinjoin. In: Data privacy management, cryptocurrencies and blockchain technology. Springer, pp 421–436
11. Ermilov D, Panov M, Yanovich Y (2017) Automatic bitcoin address clustering. In: 2017 16th IEEE International conference on machine learning and applications (ICMLA). IEEE, pp 461–466
12. Fleder M, Kester MS, Pillai S (2015) Bitcoin transaction graph analysis. arXiv:1502.01657
13. Ivgi N (2019) Blockstream - esplora - privacy-analysis.js. https://github.com/Blockstream/esplora/blob/master/client/src/lib/privacy-analysis.js
14. Maksutov AA, Alexeev MS, Fedorova NO, Andreev DA (2019) Detection of blockchain transactions used in blockchain mixer of coin join type. In: 2019 IEEE Conference of Russian young researchers in electrical and electronic engineering (EIConRus). IEEE, pp 274–277
15. Maurer FK, Neudecker T, Florian M (2017) Anonymous coinjoin transactions with arbitrary values. In: 2017 IEEE Trustcom/BigDataSE/ICESS. IEEE, pp 522–529
16. Maxwell G (2013) Coinjoin: Bitcoin privacy for the real world. https://bitcointalk.org/?topic=279249
17. Maxwell G (2013) I taint rich! (raw txn fun and disrupting 'taint' analysis;>51kbtc linked!) https://bitcointalk.org/?topic=139581
18. Maxwell G (2013) Really really ultimate blockchain compression: Coinwitness https://bitcointalk.org/index.php?topic=277389
19. Meiklejohn S, Orlandi C (2015) Privacy-enhancing overlays in bitcoin. In: International conference on financial cryptography and data security. Springer, pp 127–141
20. Meiklejohn S, Pomarole M, Jordan G, Levchenko K, McCoy D, Voelker GM, Savage S (2013) A fistful of bitcoins: characterizing payments among men with no names. In: Proceedings of the 2013 conference on internet measurement conference, pp 127–140
21. Nakamoto S (2008) Bitcoin: a peer-to-peer electronic cash system. bitcoin.org
22. Ober M, Katzenbeisser S, Hamacher K (2013) Structure and anonymity of the bitcoin transaction graph. Future Internet 5(2):237–250
23. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, et al. (2019) Pytorch: An imperative style, high-performance deep learning library. arXiv:1912.01703
24. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, et al. (2011) Scikit-learn: Machine learning in python. J Mach Learn Res 12:2825–2830
25. Bitcoin Wiki contributors (2013) Bitcoin WIKI: Shared coin. https://en.bitcoin.it/wiki/Shared_coin last accessed: January 23, 2020
26. Reid F, Harrigan M (2013) An analysis of anonymity in the bitcoin system. In: Security and privacy in social networks. Springer, pp 197–223
27. Ron D, Shamir A (2013) Quantitative analysis of the full bitcoin transaction graph. In: International conference on financial cryptography and data security. Springer, pp 6–24
28. Ruffing T, Moreno-Sanchez P, Kate A (2014) Coinshuffle: Practical decentralized coin mixing for bitcoin. In: European symposium on research in computer security. Springer, pp 345–364
29. Sáez M (2020) Blockchain-enabled platforms: Challenges and recommendations. Int J Interactive Multimed Artif Intell 6(3)
30. ShenTu Q, Yu J (2015) Research on anonymization and de-anonymization in the bitcoin system. arXiv:1510.07782
31. Southurst J (2014) Blockchain's sharedcoin users can be identified, says security expert. https://www.coindesk.com/blockchains-sharedcoin-users-can-identified-says-security-expert
32. Spagnuolo M, Maggi F, Zanero S (2014) Bitiodine: Extracting intelligence from the bitcoin network. In: International conference on financial cryptography and data security. Springer, pp 457–468

33. Tang J, Qu M, Wang M, Zhang M, Yan J, Mei Q (2015) Line: Large-scale information network embedding. In: Proceedings of the 24th international conference on world wide web, pp 1067–1077
34. Valenta L, Rowan B (2015) Blindcoin: Blinded, accountable mixes for bitcoin. In: International conference on financial cryptography and data security. Springer, pp 112–126
35. Weber M, Domeniconi G, Chen J, Weidele DKI, Bellei C, Robinson T, Leiserson CE (2019) Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. arXiv:1908.02591
36. Wu J, Yuan Q, Lin D, You W, Chen W, Chen C, Zheng Z (2020) Who are the phishers? phishing scam detection on ethereum via network embedding. IEEE Transactions on Systems, Man, and Cybernetics: Systems
37. Zheng B, Zhu L, Shen M, Du X, Guizani M (2020) Identifying the vulnerabilities of bitcoin anonymous mechanism based on address clustering. Science China Information Sciences 63(3): 1–15
38. Ziegeldorf JH, Grossmann F, Henze M, Inden N, Wehrle K (2015) Coinparty: Secure multi-party mixing of bitcoins. In: Proceedings of the 5th ACM conference on data and application security and privacy, pp 75–86
39. Ziegeldorf JH, Matzutt R, Henze M, Grossmann F, Wehrle K (2018) Secure and anonymous decentralized bitcoin mixing. Futur Gener Comput Syst 80:448–466

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Xiaowen Sun received the bachelor's degree in networking engineering from Qingdao University, in 2017, he is currently pursuing the master's degree in computer science and technology in the Beijing University of Posts and Telecommunications. His research interests include blockchain, deep learning, and natural language processing.

Tan Yang received the Ph.D. degree from the Beijing University of Posts and Telecommunications, in 2010. She is currently an Associate Professor with the Beijing University of Posts and Telecommunications. Her research interests include network performance evaluation, mobile internet and blockchain system.

Bo Hu received his PhD degree of communications and information systems in 2006, from Beijing University of Posts and Telecommunications (BUPT), China. Currently, he is a professor in the State Key Laboratory of Networking and Switching Technology, BUPT. He has published over 80 research papers in international journals & conferences and co-authored 3 books. He is active in the ITU-T SG13 for international standards and focus on the mobility management and machine learning for IMT-2020(5G) & beyond. His research interests include: integrated satellite and terrestrial mobile communication system for B5G and 6G, network artificial intelligence and mobility management & control.