Exploring Unconfirmed Transactions for Effective Bitcoin Address Clustering

Kai Wang
wangk20@fudan.edu.cn
Fudan University

Yakun Cheng
ykcheng22@m.fudan.edu.cn
Fudan University

Michael Wen Tong
mktong21@m.fudan.edu.cn
Fudan University

Zhenghao Niu
zhniu23@m.fudan.edu.cn
Fudan University

Jun Pang
jun.pang@uni.lu
University of Luxembourg

Weili Han
wlhan@fudan.edu.cn
Fudan University

ABSTRACT

The advancement of clustering heuristics has demonstrated that the addresses of Bitcoin, which are protected by their anonymous mechanisms, can be de-anonymized. While the state-of-the-art (SOTA) clustering heuristics focus on confirmed transactions stored in the blockchain, they ignore unconfirmed transactions in the mempool. These unconfirmed transactions contain information about transactions before being stored in the blockchain, covering additional address associations that can improve Bitcoin address clustering.

In this paper, we bridge the gap by combining confirmed and unconfirmed transactions for effective Bitcoin address clustering. First, we introduce a reliable data collection framework to collect both confirmed and unconfirmed Bitcoin transactions. Second, we propose two novel clustering heuristics that exploit specific behavior patterns in unconfirmed transactions and uncover additional address associations. Finally, we construct a labeled dataset and experimentally show that the effectiveness of our proposed clustering heuristics, improving recall by at least three times with higher precision compared to the SOTA clustering heuristics. Our findings show the value of unconfirmed transactions for Bitcoin address clustering and further reveal the challenges of achieving anonymity in Bitcoin. To the best of our knowledge, our study is the first to explore unconfirmed transactions for Bitcoin address clustering.

CCS CONCEPTS

• **Security and privacy → Pseudonymity, anonymity and untraceability**.

KEYWORDS

Bitcoin, address clustering, unconfirmed transactions

ACM Reference Format:
Kai Wang, Yakun Cheng, Michael Wen Tong, Zhenghao Niu, Jun Pang, and Weili Han. 2024. Exploring Unconfirmed Transactions for Effective Bitcoin Address Clustering. In *Proceedings of the ACM Web Conference 2024 (WWW '24), May 13–17, 2024, Singapore, Singapore.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3589334.3645684

WWW '24, May 13–17, 2024, Singapore, Singapore

ACM ISBN 979-8-4007-0171-9/24/05
https://doi.org/10.1145/3589334.3645684

1 INTRODUCTION

Introduced in 2008, Bitcoin [28] provides a pseudo-anonymous payment system that tries to decouple a user's real identity from his Bitcoin addresses. Bitcoin utilizes the blockchain as the distributed ledger. In Bitcoin, when a transaction is initiated, it is stored in a temporary storage of *unconfirmed* transactions, commonly referred to as the mempool. An *unconfirmed* transaction becomes a *confirmed* transaction only when stored in a confirmed block of the blockchain. Notably, a new confirmed transaction is considered irreversible when the blockchain receives five confirmed blocks after the transaction [4]. Furthermore, some unconfirmed transactions may become *failed* transactions with no chance of being confirmed.

In Bitcoin transactions, users can generate an unlimited number of new addresses for various transactions to hide their real identities. The pseudo-anonymous ecosystem of Bitcoin has attracted an increasing number of users, including criminals who leverage Bitcoin to obfuscate their real identities during the transfer of illicit funds. To deal with such criminal activities in Bitcoin, a large number of studies focus on Bitcoin de-anonymization. Central to this domain is the notion of Bitcoin address clustering [48], which aims to identify multiple addresses controlled by the same entity. At present, the clustering heuristic stands as the predominant method for Bitcoin address clustering, which achieves address clustering by analyzing behavior patterns in confirmed transactions [17, 18, 26, 35, 39, 41]. For instance, transactions with multiple inputs usually arise when a user lacks an Unspent Transaction Output (UTXO) that has sufficient bitcoins to cover the payment. One straightforward idea, known as the co-spend heuristic [17, 26, 39], considers that all input addresses in a Bitcoin transaction belong to the same entity. In practice, clustering heuristics find extensive application in various domains, including case investigations [5, 9, 30] and the tracking of illicit funds [13, 22, 41], particularly within firms specializing in blockchain data analytics, such as Chainalysis [3].

However, the state-of-the-art (SOTA) clustering heuristics focus only on confirmed transactions but ignore unconfirmed transactions, leading to numerous undiscovered address associations. On the one hand, some unconfirmed transactions will not be stored in the blockchain if they become failed transactions. Consequently, the SOTA clustering heuristics miss potentially valuable address associations hidden in these failed transactions. On the other hand, unconfirmed transactions can provide important insights into the

state of transactions before being stored in the blockchain. For example, to incentivize miners to store a user's unconfirmed transaction in the blockchain more quickly, the user may initiate a new transaction with a high fee that spends the UTXO(s) of the unconfirmed transaction. This behavior forms a dependency chain in the mempool that contains address associations. However, the dependencies among transactions are not stored in the blockchain later. Thus, focusing only on confirmed transactions cannot capture such insight.

In this paper, we present a practical approach for improving Bitcoin address clustering, leveraging both confirmed and unconfirmed transactions. First, we introduce a reliable data collection framework, including two sub-components: Confirmed Transaction Collector (CTC) and Unconfirmed Transaction Processor (UTP). Hereby, CTC, which utilizes a single node running a Bitcoin client *Bitcoin Core*[1], is responsible for copying confirmed transactions in the blockchain. UTP is responsible for recording and processing unconfirmed transactions in real time. It comprises five nodes, each of which runs a modified Bitcoin Core. Subsequently, we propose two novel clustering heuristics specifically designed for unconfirmed transactions, leveraging the Replace-by-fee (RBF) proposed by Bitcoin Improvement Proposal (BIP)125 [11] and the unconfirmed transaction dependency chain mentioned in BIP141 [24]. To validate our approach, we construct a labeled dataset based on Bitcoin ordinal inscriptions [36] and demonstrate that our approach improves the recall by at least three times with higher precision compared to the SOTA clustering heuristics. The increase in recall indicates that our approach uncovers additional address associations, thus reducing entities incorrectly clustered. Furthermore, we show that our approach reduces the number of entities in the clustering results of the SOTA clustering heuristics by at least 20.28%, which can reduce the error of addresses that should belong to the same entity, but being clustered into multiple entities. Finally, we find that unconfirmed transactions have a greater impact on the clustering results for future periods than those from past periods.

To the best of our knowledge, our study is the first to explore unconfirmed transactions to cluster addresses in Bitcoin. In summary, our main contributions in this paper are threefold:

- **Novel heuristics**: We propose two novel clustering heuristics to uncover additional address associations by analyzing the specific behavior patterns in unconfirmed transactions, in order to improve Bitcoin address clustering. Experimental results show that our proposed clustering heuristics can effectively utilize unconfirmed transactions to uncover address associations, significantly improving recall by at least three times with higher precision.
- **Data collection**: We introduce a reliable data collection framework to record and process unconfirmed transactions in Bitcoin in real time. We release a part of the dataset[2] as a benchmark for future studies.
- **Labeling method**: We present a method for constructing a labeled dataset based on Bitcoin ordinal inscriptions. This method addresses, to some extent, the critical issue in the field of Bitcoin address clustering, i.e., the lack of labeled datasets to validate clustering results. In this paper, we construct and release a dataset[3] encompassing 20 entities and 62,971 addresses.

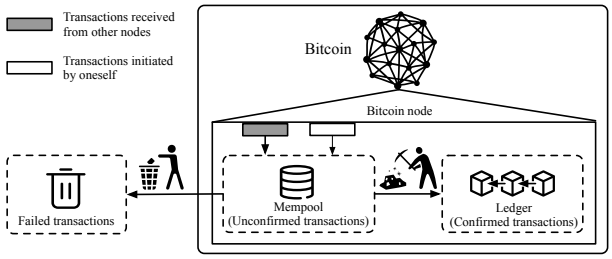

Figure 1: Life-cycle of a Bitcoin transaction.

2 BACKGROUND

2.1 Mempool in a Bitcoin Node

Bitcoin is established on a set of Bitcoin nodes, each of which stores a ledger of confirmed transactions. Bitcoin blockchain acts as the distributed ledger.

As shown in Figure 1, before a transaction is stored in the blockchain, it (as an *unconfirmed* transaction) is temporarily stored in the mempool of a node. The node will validate the transaction that is from other nodes or initiated by itself based on established criteria, such as signature validity. If the transaction meets the established criteria, the node will add the transaction to its mempool. Miners select transactions from their own mempools and then store these transactions in a block. They compete to append the block to the blockchain through proof-of-work. The winning miner propagates his block to other nodes, while other nodes append the block to their own ledgers. When the block is appended to the ledgers of all nodes, it indicates that the block is appended to the blockchain. Then, transactions in the block become *confirmed* transactions. Note that some unconfirmed transactions might never get confirmed due to, e.g., paying insufficient fees. These transactions are considered as *failed* transactions.

Each unconfirmed transaction in the mempool has some additional fields, e.g., *replaceable*, *time*, *depends*, and *spentby*, which are not present in confirmed transactions. The latter three fields are exclusively available when a transaction is in Bitcoin mempool and disappear once the transaction is confirmed. (1) The field *replaceable* is a Boolean value, indicating whether this transaction can be replaced by another transaction. (2) The field *time* annotates the moment at which the transaction enters a particular node's mempool that may exhibit minor variances across different mempools. (3) The field *depends* of a transaction records unconfirmed transactions whose UTXO(s) is spent by this transaction. (4) The field *spentby* of a transaction records unconfirmed transactions spending outputs of this transaction. These fields contain rich information about a transaction before it is confirmed, which can be utilized in our study for Bitcoin address clustering.

[1] https://bitcoin.org/en/releases/22.0/

[2] See details at https://drive.google.com/drive/folders/1Vc5p9qro8zh6lV6lLqQMB4AtSvdhLjiT?usp=sharing

[3] See details at https://github.com/UnconfirmedTransactions/LabeledDataset

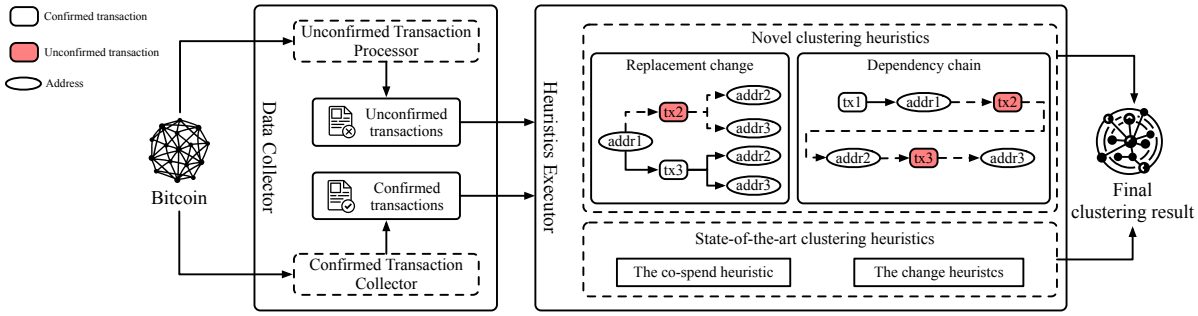

Figure 2: Our approach for Bitcoin address clustering by combining unconfirmed transactions and confirmed transactions.

2.2 Bitcoin Address Clustering

The SOTA clustering heuristics rely on behavior patterns in confirmed transactions to uncover address associations. For example, the co-spend heuristic considers that all inputs of a transaction are controlled by the same entity. In practice, however, a few studies [13, 17, 18] advocate the exclusion of Coinjoin transactions [25] before applying the co-spend heuristic. This is mainly because Coinjoin transactions employ a trustless method for combining multiple Bitcoin payments into a single transaction, thereby obfuscating the relationship between senders and recipients.

In addition, various heuristics, known as the change heuristics, have been introduced in previous studies [1, 6, 7, 18, 26]. In Bitcoin, a UTXO represents a certain amount of bitcoins. It is an indivisible unit and must be fully spent in a transaction. This requires senders to use a change address to receive the remaining bitcoins. Thus, the address of the sender (the input address) and the change address should be controlled by the same entity. For such heuristics, it is key to identify change addresses in transactions.

So far, the SOTA clustering heuristics focus only on behavior patterns in confirmed transactions, but ignore the additional information of unconfirmed transactions.

3 APPROACH

3.1 Overview

As shown in Figure 2, our approach consists of two components: Data Collector and Heuristics Executor. Data Collector contains two sub-components: Confirmed Transaction Collector (CTC) and Unconfirmed Transaction Processor (UTP). CTC copies confirmed transactions from the ledger of a node, while UTP collects and processes unconfirmed transactions. Data Collector subsequently transfers both confirmed and unconfirmed transactions to Heuristics Executor. Then, Heuristics Executor, consisting of our proposed clustering heuristics and the SOTA clustering heuristics, clusters Bitcoin addresses.

3.2 Data Collector

CTC. As shown in Figure 2, CTC with one node running a Bitcoin client, referred to as Bitcoin Core, copies confirmed transactions from the ledger of the node. As a consequence of the Taproot upgrade of Bitcoin, a new address type known as Taproot address has been introduced [44]. Notably, BlockSci [17], a widely used Bitcoin transaction parsing tool, is unable to parse this address

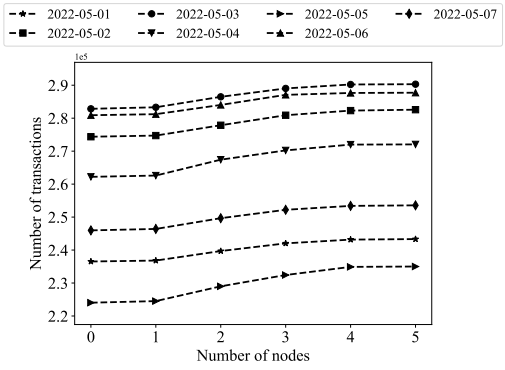

Figure 3: The number of transactions collected vs. the number of nodes running a modified Bitcoin core.

type. We optimize BlockSci as BlockSci-modified[4], which can parse Taproot addresses, thereby ensuring the integrity of our dataset.

UTP. UTP consists of a set (five in this paper) of nodes, each of which runs a modified Bitcoin Core. It aims to collect as many unconfirmed transactions as possible in Bitcoin. Here, UTP tries to solve two key issues as follows:

(1) *How can UTP collect unconfirmed transactions in real time?* When using the Remote Procedure Call interface provided by Bitcoin Core to collect unconfirmed transactions, the first call has to obtain hashes of unconfirmed transactions in the mempool presently. Then, the second call retrieves detailed transactions based on these hashes. Due to the time gap between these two calls, part of the transactions may be removed from the mempool, resulting in missing these removed transactions. To collect unconfirmed transactions in real time, we modify Bitcoin Core. Our modified Bitcoin Core[5] monitors the arrival of all transactions and record them in real time.

(2) *How can UTP collect unconfirmed transactions in Bitcoin as many as possible?* Due to multiple factors such as the decentralized network of Bitcoin, network latency, and bandwidth limitations, unconfirmed transactions received by different nodes might vary. To achieve a comprehensive collection of unconfirmed transactions in Bitcoin, we perform experiments to evaluate the completeness of

[4]See details at https://github.com/UnconfirmedTransactions/BlockSci-modified
[5]See details at https://github.com/UnconfirmedTransactions/BitcoinCore-modified

unconfirmed transactions collected by UTP. We increase the number of nodes and measure the number of deduplicated unconfirmed transactions collected per day from May 1, 2022 to May 7, 2022. Figure 3 shows that the number of deduplicated unconfirmed transactions rarely increases when the number of nodes reaches five. Similarly, as shown in Appendix A, the total number of unconfirmed transactions shows almost no increase when the number of nodes exceeds 5. That is, UTP can collect approximately all unconfirmed transactions in Bitcoin when deploying five nodes. Therefore, we deploy five nodes, each of which runs a modified Bitcoin Core, to collect as many unconfirmed transactions as possible in Bitcoin.

As shown in Appendix B, there are some extra fields of an unconfirmed transaction in the mempool. In this paper, we focus on seven fields, i.e., *fee*, *vsize*, *time*, *removetime*, *depends*, *spentby*, and *replaceable*, which are relevant to subsequent analysis of behavior patterns in unconfirmed transactions.

Finally, we build a mempool state database for the mempools of five nodes. In the database, we set *time* and *removetime* as indexes for each unconfirmed transaction. Given a specific time, the database is able to retrieve every unconfirmed transaction in each mempool at the moment. Note that a transaction output may be spent by multiple unconfirmed transactions. We make two adjustments to the original transaction structure. Specifically, the first field *output.is_spent*, a Boolean type, indicates whether the *output* has been spent by transactions. The second field *output.spent_tx* is a list containing hashes of transactions that spend this *output*.

Since failed transactions are removed from the mempool and no longer exist in Bitcoin, we can simply identify failed transactions by excluding confirmed transactions from unconfirmed transactions.

3.3 Novel Clustering Heuristics

We explore two mechanisms in unconfirmed transactions to design novel clustering heuristics.

(1) Replace-by-fee (RBF) [11]. It allows a sender to replace their unconfirmed transaction by initiating another transaction that pays a higher fee. Due to the limitation of fixed block size, miners give priority to transactions with a higher feerate (*fee*/*vsize*) to maximize their profit. Note that a transaction can only be replaced when the field *replaceable* of the transaction is set to true.

(2) Unconfirmed transaction dependency chain [24]. The Bitcoin mempool is designed to accept unconfirmed transactions that spend UTXO(s) of other unconfirmed transactions. The user can initiate a new transaction to spend UTXO(s) of his unconfirmed transactions. As a result, it is common to form an unconfirmed transaction dependency chain in the mempool. In a dependency chain, each unconfirmed transaction spends UTXO(s) of the preceding unconfirmed transaction, which in turn spends UTXO(s) of another unconfirmed transaction, and so forth. Dependency chains typically form for two reasons. One is that users want miners to store their multiple transactions in the blockchain at one time, without waiting for a transaction to be confirmed before initiating a new one that spends UTXO(s) of the transaction. The other reason is related to a transaction pattern known as Child-Pays-for-Parent (CPFP). To incentivize miners to store the parent transaction of a user in the blockchain early, the user can initiate a child transaction that pays a high fee and spends UTXO(s) of the parent transaction.

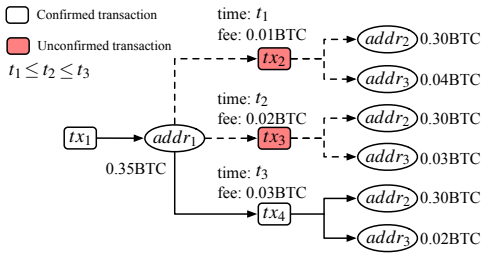

Figure 4: Example of *replacement change* heuristic.

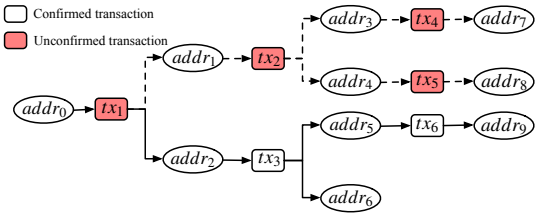

Figure 5: Example of *dependency chain* heuristic.

For the first mechanism, we design a clustering heuristic, *replacement change* shown in Figure 2, to identify the change address.

***Replacement Change* heuristic.** Let T denote a set of transactions, defined as $T = \{tx_1, tx_2, \ldots, tx_n\}$ with $n \geq 2$. We identify the change address if (1) the field *replaceable* of each transaction in T is *True*; (2) transactions in T spend the same UTXO; (3) fees of transactions in T increase as the field *time* increases; and (4) the address appears in the output of each transaction in T and the amounts received by the address decrease as the field *time* increases.

This heuristic works for the following reason. In real-world trade of goods, the price of goods is typically negotiated between two parties and does not change arbitrarily. When the sender increases the fee, the amount paid to the recipient remains the same, while the amount received by the change address decreases. As shown in Figure 4, tx_2, tx_3, and tx_4 all attempt to spend the 0.35 BTC in $addr_1$. When the fee increases, the amount received by $addr_2$ remains the same, and the amount received by $addr_3$ decreases. Therefore, we can identify $addr_2$, which consistently receives the same amount, as the recipient address, and identify $addr_3$, which receives a gradually smaller amount, as the sender's change address.

For the second mechanism, we design another clustering heuristic, *dependency chain* shown in Figure 2, for the unconfirmed transaction dependency chain.

***Dependency Chain* heuristic.** Let T denote a sequence of unconfirmed transactions, defined as $T = \langle tx_1, tx_2, \ldots, tx_n \rangle$ with $n \geq 2$. Each transaction tx_k in the range $2 \leq k \leq n$ spends partial outputs of the previous transaction tx_{k-1}, denoted as O_{k-1}. Considering that tx_1 is the first transaction and spends outputs of confirmed transactions, we use O_0 to represent all inputs of tx_1. For every k in the range $2 \leq k \leq n$, if (1) the hash of tx_k is in the field *spentby* of tx_{k-1}; or (2) the hash of tx_{k-1} is in the field *depends* of tx_k, all transactions in T are initiated by one same entity, i.e., $\bigcup_{k=0}^{n} O_k$ are all controlled by the same entity.

This heuristic works for two reasons. First, in the design of popular Bitcoin wallets like Binance, Coinbase, Electrum, and BlueWallet, users can view and spend UTXO(s) of unconfirmed transactions initiated by themselves. To protect funds for users, these wallets do not allow users to view and spend UTXO(s) sent to them in the unconfirmed transactions initiated by others. Second, a Bitcoin transaction is considered irreversible when it is stored in the block and then five new blocks are added to the blockchain. For commercial trades, users do not use unconfirmed transactions as an indicator of fund arrival or spend UTXO(s) of unconfirmed transactions. Instead, they have to wait for new blocks to ensure transaction security and irreversibility. As a result, transactions between different entities do not form an unconfirmed transaction dependency chain.

As shown in Figure 5, we can identify two dependency chains: $\langle tx_1, tx_2, tx_4 \rangle$ and $\langle tx_1, tx_2, tx_5 \rangle$, while the three transactions tx_1, tx_3, and tx_6, as well as the two transactions tx_1 and tx_3, do not form a dependency chain. Applying this new heuristic, we can cluster $\{addr_0, addr_1, addr_3\}$ into the first entity and $\{addr_0, addr_1, addr_4\}$ into a second entity. These two entities can be further merged into a larger entity due to the common address $addr_0$ in both clusters.

Note that our proposed heuristics aim to uncover address associations that are beyond the scope of the SOTA clustering heuristics, rather than replacing them. The final clustering result is the combination of the results of the SOTA clustering heuristics and our proposed heuristics.

4 EVALUATION

To evaluate the effectiveness of our approach, this section presents the experimental results to address three key issues as follows:

(1) **Clustering result validation.** Is the clustering result of our approach accurate, and does our approach possess the capability to uncover additional address associations? (Section 4.3)
(2) **Impact measurement.** How much impact does our approach have on the clustering results of the SOTA clustering heuristics? (Section 4.4)
(3) **Temporal analysis.** What pattern does the impact of our approach show across different periods? (Section 4.5)

4.1 Dataset

We collect a total of 116,514,258 unconfirmed transactions (from May 1, 2022 to May 31, 2023). Among these, 113,296,795 (97.24%) unconfirmed transactions become confirmed transactions, while the rest of them (2.76%) become failed transactions. These confirmed transactions involve 179,352,220 Bitcoin addresses, while the failed transactions involve 12,366,745 Bitcoin addresses, 843,892 of which are not recorded in any confirmed transactions. Two case studies in Appendix C show the presence of behavior patterns in unconfirmed transactions, while not present in confirmed transactions.

Among all unconfirmed transactions, 44,558,888 (38.24%) have the field *replaceable* set to true, indicating their potential for replacement by other transactions. Within this subset, 3,217,463 (2.76%) are actually replaced and end up as failed transactions. Additionally, 26,060,639 transactions (22.37%) have non-null values for the field *depends* or *spentby*, indicating the formation of dependency chains.

4.2 Baseline

As shown in Figure 2, we employ six SOTA clustering heuristics as the baseline.

Co-spend (short for CS) considers all inputs of a transaction are controlled by the same entity if the transaction is not a Coinjoin transaction. We use the algorithm developed by Goldfeder *et al.* [7] to determine whether a transaction is a Coinjoin transaction.
Androulaki *et al.* (short for A) [1] identify the change address of a transaction sender if (1) the transaction must have exactly two outputs; and (2) the address is the only *fresh* address in the outputs, meaning that it has not been previously used in the blockchain.
Meiklejohn *et al.* (short for M) [26] identify the change address of a transaction sender if (1) the transaction is not a coinbase transaction; (2) the address is the only *fresh* address in the outputs; and (3) there is no address used as both an input and an output in this transaction.
Goldfeder *et al.* (short for G) [7] utilize the criteria established by Meiklejohn *et al.* [26], but they also add a further condition: (4) the transaction cannot be a Coinjoin transaction.
Ermilov *et al.* (short for E) [6] identify the change address of a transaction sender if (1) the number of inputs is not two; (2) the transaction has exactly two outputs; (3) there is no address used both as an input and an output in the transaction; (4) the address is the only *fresh* address in outputs; and (5) the amount received by the address is precise to a minimum of four decimal places.
Kappos *et al.* (short for K) [18] identify the change address of a transaction sender if (1) the transaction is a node in a *peel chain*; (2) the amount received by the address is spent and the spent transaction is also a node in the *peel chain*.

To describe experimental results clearly, we assign a name to each clustering result, composed of the heuristic and the state of transactions. The SC clustering result refers to the result of applying one of the SOTA clustering heuristics to confirmed transactions. The SF clustering result refers to the result of applying one of the SOTA clustering heuristics to failed transactions. The NU clustering result refers to the result of applying our proposed clustering heuristics to unconfirmed transactions. We denote the merging of clustering results with a plus sign. For instance, SC+SF represents a merge of SC and SF clustering results under the same heuristic. In cases where no heuristic is applied, we refer to it as None, with each address being considered as an isolated entity. Note that merging the SC and SF clustering results essentially represents applying the SOTA clustering heuristics to unconfirmed transactions.

4.3 Clustering Result Validation

To validate clustering results, we construct a labeled dataset and analyze the clustering results from multiple metrics.

Labeling method. Validating clustering results requires the availability of labeled datasets. However, there is no publicly available labeled dataset since our dataset is relatively recent. Thus, we propose a labeling method to validate our clustering results.

The labeling method is based on Bitcoin ordinal inscriptions [42]. Bitcoin ordinal inscriptions are digital assets created by attaching information to an individual satoshi, the smallest denomination in Bitcoin, through the Ordinals protocol [36]. Two features are worth noting in this protocol. First, creating an individual ordinal

Table 1: Number of addresses per collection that we collect.

Name	Number	name	Number
DogePunks	11,596	Battle of BTC	11,118
BTC Virus	9,993	Bixels	10,024
Bitcoin Crypto DickButts	8,195	Mesh Beatles	3,305
OrdiRats	2,317	Bsos	1,819
420 Rabbits	1,192	Block Gods	1,049
Taproot Cows	557	Pixel Panda Wars	399
STARBREEDER	374	bitCroSkull	334
Cubic A: Kaz Marquis	196	Familiar Fronks	169
Majo	110	Ordinal Cat Warriors	102
10^2 Islands	101	iDclub Pass	21

inscription must follow a two-phase procedure: a commit transaction and a reveal transaction [36]. Thus, both commit and reveal transactions are initiated by the same entity. Second, an ordinal inscription collection consists of a set of individual ordinal inscriptions. In the early stages, it is common to employ off-chain data and social consensus to establish the attribution of inscriptions to specific collections. After the parent-child inscription mechanism is introduced, it is utilized to create collections, with child inscriptions being created exclusively by the owner of the parent inscription, resulting in all children being members of the same collection [36]. Thus, all ordinal inscriptions of a collection are created by the same entity. In summary, the input addresses of both commit and reveal transactions for each inscription in a collection are controlled by the same entity (see more details in Appendix D).

The specific process of the labeling method is given as follows. First, we gather a Bitcoin ordinal inscription collection, defined as $S = \{o_1, o_2, \ldots, o_n\}$ with $n \geq 2$. For each ordinal inscription o_k in S, we identify its corresponding commit transaction ctx_k and reveal transaction rtx_k. Next, we extract the input addresses of transaction ctx_k and rtx_k in the range $1 \leq k \leq n$, denoted as I_k. Finally, we consider $\bigcup_{k=0}^{n} I_k$ are controlled by the same entity.

For this paper, we sample collections based on the creation time and the size, ensuring that our dataset can reflect the entire ordinal inscription ecosystem. As a result, we gather 20 collections (entities) from the website [31] and label 62,971 addresses as the validation dataset, as shown in Table 1. The creation time of these collections spans each month from February 2023 to May 2023. Besides, these collections encompass a variable quantity of addresses, ranging from several dozen to over ten thousand.

Validation metrics. We measure clustering results from two aspects. First, we show the number of entities successfully identified (N). Second, we evaluate the quality of addresses in each identified entity through four metrics: Precision (P), Recall (R), Weighted Precision (WP), and Weighted Recall (WR). The first two metrics are commonly used in the study [2], while the last two metrics are introduced in the study [41]. The definitions of these four metrics are as follows, where m denotes the total number of entities, i.e., 20, and E_i denotes ith entity. Addresses of E_i are clustered into n clusters, with c_{ij} representing the jth cluster of E_i. We denote the union of these clusters as C_i. We use the set v_{ij} to denote the addresses of Ei that are clustered into the cluster c_{ij}. We denote the union of v_{ij} as V_i. w_{ij} represents the proportion of the set c_{ij} within entity E_i. The greater the number of addresses within a cluster, the more accurately it reflects the characteristics of the entity and, thus, the higher its significance in the clustering results.

Table 2: Comparison between our proposed clustering heuristics and SOTA clustering heuristics.

Heuristics	N	P(%)	R(%)	WP(%)	WR(%)
CS	8	0.09	**4.47**	7.37	2.81
CS+NU	18	94.27	**74.32**	42.23	18.73
CS+A	10	28.49	**3.93**	14.11	2.85
CS+A+NU	16	94.49	**59.58**	35.53	18.24
CS+M	12	55.29	**11.84**	11.75	0.24
CS+M+NU	15	90.29	**52.04**	27.89	6.23
CS+G	12	55.29	**11.84**	11.75	0.24
CS+G+NU	15	90.29	**52.04**	27.89	6.23
CS+E	9	35.43	**4.96**	11.96	2.81
CS+E+NU	18	94.58	**73.87**	42.15	18.74
CS+K	12	75.21	**12.46**	12.52	8.32
CS+K+NU	16	90.43	**77.53**	27.72	16.28

$$P = \frac{\sum_{i=1}^{m} |V_i|}{\sum_{i=1}^{m} |C_i|}, \quad R = \frac{\sum_{i=1}^{m} |V_i|}{\sum_{i=1}^{m} |E_i|}$$

$$WP = \frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} \frac{|v_{ij}|}{|C_i|}, \quad WR = \frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} \frac{|v_{ij}|}{|E_i|}$$

$$\text{where } v_{ij} = E_i \cap c_{ij},\ V_i = \bigcup_{j=1}^{n} v_{ij},\ C_i = \bigcup_{j=1}^{n} c_{ij},\ w_{ij} = \frac{|c_{ij}|}{|C_i|}.$$

Validation results. Table 2 demonstrates the effectiveness of our proposed clustering heuristics compared to the SOTA clustering heuristics. Our proposed clustering heuristics identify more entities while achieving a precision of over 90%. Notably, our proposed heuristic significantly improves recall. Even in the smallest improvement cases CS+M and CS+G, our proposed clustering heuristics still improve recall by three times with higher precision. This indicates that our proposed clustering heuristics can uncover many additional address associations that are beyond the scope of the SOTA clustering heuristics. Both weighted precision and weighted recall exhibit a significant improvement, further showing our proposed clustering heuristics can uncover additional address associations and identify more addresses belonging to the same entity. Significantly, the results for both CS+M and CS+G are identical. This is because CS already achieves the exclusion of Coinjoin transactions, which is the sole distinction between CS+G and CS+M.

4.4 Impact Measurement

Building upon the effectiveness of our proposed clustering heuristics, we measure their impact on the SC clustering results.

Settings. We apply the SOTA clustering heuristics to failed transactions and our proposed clustering heuristics to unconfirmed transactions to uncover additional address associations. When no clustering heuristic is applied, we consider each address in the confirmed transactions as an isolated entity, i.e., 179,352,220 isolated entities.

Measurement metric. We employ the reduced number of entities in the clustering results as the metric to measure the impact of our approach. Suppose the SOTA clustering heuristics cluster addresses of an entity into n clusters, denoted as $\{C_1, C_2, \ldots, C_n\}$ with $n \geq 2$. Our proposed clustering heuristics produce an additional cluster, denoted as C_{n+1}. If $C_{n+1} \cap C_i \neq \emptyset$ for i in the range $1 \leq i \leq n$, C_{n+1} can merge these n clusters, reducing the number of entities.

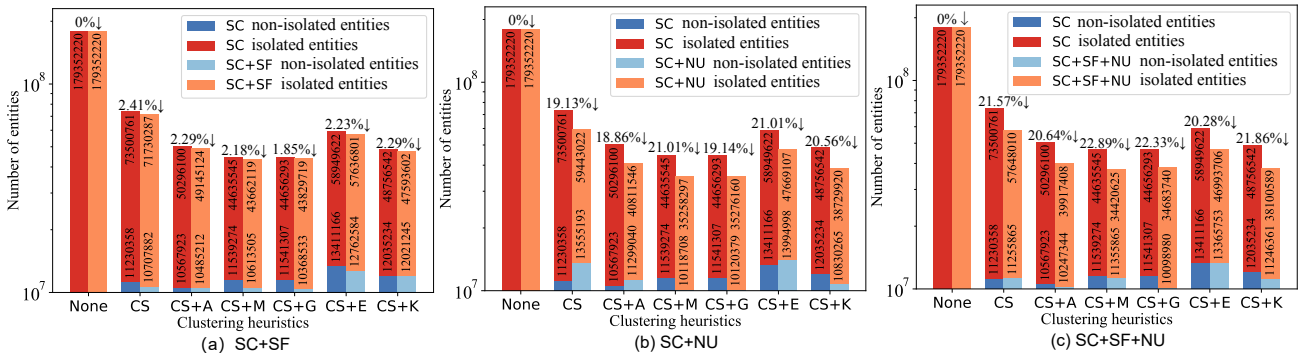

Figure 6: Comparison of the number of entities in the SC clustering result and other three clustering results. While an isolated entity contains only one Bitcoin address, a non-isolated entity contains multiple Bitcoin addresses.

Note that C_i may contain only one Bitcoin address for i in the range $1 \leq i \leq n$. Hence, the reduction in the number of entities reflects the ability of our approach to reduce errors where addresses belonging to the same entity are clustered into multiple entities.

Measurement results. Figure 6(a) shows the impact of failed transactions on the SC clustering results. The SC+SF clustering result has a reduced number of entities compared to the SC clustering result across various SOTA clustering heuristics. Notably, the most significant reduction is observed in the result of the CS heuristic, with a reduction of 1,770,474 entities (2.41% of the total entities). This highlights that failed transactions contain additional address associations that are currently ignored.

Figure 6(b) shows the impact of our proposed clustering heuristics on the SC clustering results. The SC+NU clustering result has significantly fewer entities than the SC clustering result across various clustering heuristics. Notably, the CS+M clustering result is the most affected, with a reduction of 9,377,248 entities (21.01% of the total entities). This indicates that our proposed clustering heuristics reveal numerous address associations in unconfirmed transactions that are beyond the scope of the SOTA clustering heuristics.

Figure 6(c) shows the comprehensive impact of both failed transactions and our proposed clustering heuristics on the SC clustering results. The results indicate that a part of the SF clustering results and NU clustering results exhibit no overlap, further reducing the number of entities. The CS+M clustering result is the most affected, with a reduction of 10,214,920 entities (22.89% of the total entities). Our approach utilizes failed transactions and our proposed clustering heuristics to uncover numerous additional address associations, significantly improving the SC clustering results.

Appendix E shows the individual contributions of the two proposed clustering heuristics to the improvement of clustering results, revealing that the *dependency chain* heuristic significantly contributes to the improvement.

4.5 Temporal Analysis

Considering that the state of a transaction changes over time, our approach has varying degrees of impact on the SC clustering results across different periods. Hence, we conduct a monthly temporal analysis of our approach's impact on the SC clustering results in different periods, according to the reduced number of entities.

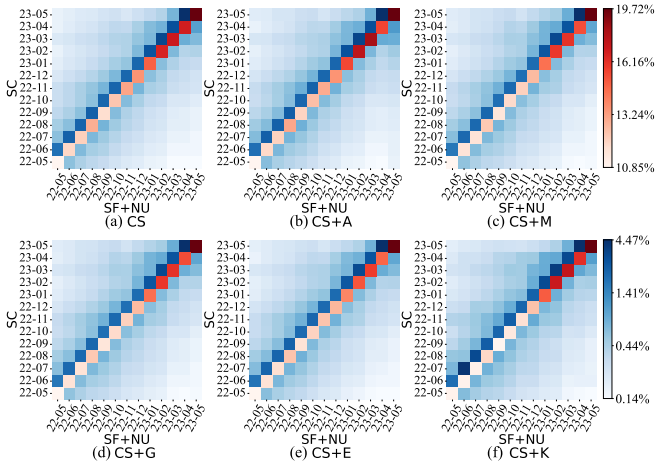

Figure 7: Temporal impact of our approach on the SC clustering results.

First, as shown in Figure 7, our approach has the most significant impact on the SC clustering results in the current month, and the impact on other months decreases month by month. Notably, we observe that the clustering results of our approach in the current month have a more significant impact on the SC clustering results in the subsequent month compared to the previous month. This is mainly because unconfirmed transactions collected by UTP in the current month may be confirmed in the subsequent month, thus leading to a more significant impact on the SC clustering results.

Second, as shown in the diagonal, the impact of our approach exhibits a growing trend over time, indicating its enduring effect. Notably, the impact experiences a significant enhancement in January 2023. Our analysis attributes this phenomenon primarily to the emergence and widespread adoption of Bitcoin ordinal inscriptions in January 2023 [36]. When users create a collection of ordinal inscriptions, they often utilize unconfirmed transaction dependency chains to create numerous ordinal inscriptions at the same time. Experimental results in Appendix F further demonstrate that the impact of our approach becomes more pronounced following the emergence of ordinal inscriptions.

5 DISCUSSION

The impact of Coinjoin transactions on clustering heuristics. Coinjoin transactions render the SOTA clustering heuristics ineffective. In this paper, our proposed clustering heuristics utilize replacement transactions and unconfirmed transaction dependency chains. To ensure timely transaction confirmation, users often avoid employing Coinjoin transactions to facilitate operations in both cases. Furthermore, the *replacement transaction* heuristic imposes very strict constraints, which reliably identify the change address.

False positive of our proposed clustering heuristics. To accelerate the confirmation of a transaction, the sender transmits the transaction hash and the corresponding UTXO to the recipient before the transaction is confirmed. The recipient then utilizes this UTXO to initiate a new transaction, forming an unconfirmed transaction dependency chain. In this dependency chain, addresses involved are not controlled by the same entity. However, this situation remains infrequent due to its potential association with unconfirmed transaction attacks, a subtype of double-spend attacks.

Labeled dataset construction. With the development of third-party platforms for Bitcoin ordinal inscriptions, users often use these platforms to create ordinal inscription collections for convenience. In this situation, a third-party platform creates multiple collections on behalf of users. Thus, all input addresses for the ordinal inscription creation transactions in these collections are controlled by the third-party platform. While this situation introduces certain imperfections into the labeled dataset, the address associations within each entity in our dataset remain accurate. Furthermore, the creation time of ordinal inscription collections in our dataset predates the development of third-party platforms. Therefore, our dataset is minimally affected by this situation.

User privacy leakage in unconfirmed transactions. The experimental results in Section 4 reveal that unconfirmed transactions can significantly reduce the anonymity of Bitcoin, but it ultimately benefits Bitcoin users by motivating further research into privacy protocols for the mempool.

6 RELATED WORK

6.1 Clustering Bitcoin Addresses

Many studies attempt to achieve de-anonymization by proposing various clustering heuristics. The SOTA clustering heuristics can generally be categorized into two main groups: the co-spend heuristic and the change heuristic. The co-spend heuristic, observed in the white paper [28], is applied in many studies [1, 21, 26, 35, 39]. Based on the co-spend heuristic, Kalodner *et al.* [17] propose to reduce clustering interference caused by Coinjoin transactions. Then, Meiklejohn *et al.* [26] and Androulaki *et al.* [1] propose the change heuristic to determine which transaction output is the address to receive change. Goldfeder *et al.* [7] and Ermilov *et al.* [6] further refine this heuristic. Kappos *et al.* [18] consider the transaction pattern *peel chain* to identify the change address. The change heuristic has been used to track illicit fund flows [13, 14, 22, 32–34, 45].

There are also studies on analyzing the effectiveness of various clustering heuristics [2, 8, 12, 23, 29, 48]. Cazabet *et al.* [2] highlight that only employing the co-spend heuristic has a relatively low recall but a high precision. Zheng *et al.* [48] demonstrate that the existing clustering heuristics do not guarantee the comprehensiveness, accuracy, and efficiency of the clustering results. Liu *et al.* [23] point out that all clustering heuristics rely on confirmed transactions stored in the blockchain.

6.2 Analyzing Bitcoin Mempool

Related studies focus on two issues: predicting the transaction confirmation time and analyzing unconfirmed transactions.

Predicting the transaction confirmation time. Many studies [10, 20, 27, 40, 46, 47] propose various methods to estimate the confirmation time. Gundlach *et al.* [10] predict the confirmation time of Bitcoin transactions by modeling the confirmation time as the time to ruin of a Cramer-Lundberg (CL) model. Ko *et al.* [20] and Zhang *et al.* [46] employ machine learning techniques to predict confirmation time of unconfirmed transactions, taking into account various factors such as block states, and mempool states.

Analyzing unconfirmed transactions. Saad *et al.* [37, 38] investigate the impact of DDoS attacks on the mempool size and the fees paid by users. Meanwhile, Dae-Yong *et al.* [19] examine the variation of unconfirmed transactions in different mempools through the Jaccard similarity index. They find that unconfirmed transactions in mempools are significantly different when a new block is produced. Kallurkar *et al.* [16] focus on statistics of failed transactions and the primary reasons for the transaction failure. Furthermore, they point out that the area of failed transactions remains unexplored. To further explore the impact of the mempool on users, we focus on the user privacy disclosed by unconfirmed transactions (including failed transactions) in the mempool, and design clustering heuristics for unconfirmed transactions.

7 CONCLUSION AND FUTURE WORK

In this paper, we present a practical approach to cluster Bitcoin addresses by combining confirmed and unconfirmed transactions, significantly improving Bitcoin address clustering. The key idea is to explore specific behavior patterns in unconfirmed transactions and propose two novel clustering heuristics for unconfirmed transactions. Then, we construct a labeled dataset based on Bitcoin ordinal inscription to validate the clustering result, and measure the impact of our approach. Experimental results reveal that our proposed clustering heuristics can uncover additional address associations and reduce the error of addresses controlled by the same entity being clustered into multiple entities.

In future, we aim to extend our analysis to other cryptocurrencies based on the UTXO model, such as Litecoin. We will also analyze the Ethereum mempool to explore the traceability of funds under the account-balance model.

ACKNOWLEDGEMENTS

This paper is supported by the National Key R&D Program of China (2022YFC3301300, 2023YFC3304400), Natural Science Foundation of China (62172100), and STCSM Key Projects (21DZ1201400). We thank all anonymous reviewers for their insightful comments. We also thank Professor Xiapu Luo for his valuable contributions and insightful suggestions. Weili Han is the corresponding author.

REFERENCES

[1] Elli Androulaki, Ghassan Karame, Marc Roeschlin, Tobias Scherer, and Srdjan Capkun. 2013. Evaluating User Privacy in Bitcoin. In *Proceedings of the 17th International Conference on Financial Cryptography and Data Security (FC) (Lecture Notes in Computer Science, Vol. 7859)*. Springer, 34–51.

[2] Rémy Cazabet, Rym Baccour, and Matthieu Latapy. 2017. Tracking Bitcoin Users Activity Using Community Detection on a Network of Weak Signals. In *Proceedings of the 6th International Conference on Complex Networks and Their Applications (Complex Networks) (Studies in Computational Intelligence, Vol. 689)*. Springer, 166–177.

[3] Chainalysis. 2022. Chainalysis: The Blockchain Data Platform. https://www.chainalysis.com/.

[4] Mauro Conti, Sandeep Kumar E, Chhagan Lal, and Sushmita Ruj. 2018. A Survey on Security and Privacy Issues of Bitcoin. *IEEE Communications Surveys & Tutorials* 20, 4 (2018), 3416–3452.

[5] Jesse Dunietz. 2017. The Imperfect Crime: How the WannaCry Hackers Could Get Nabbed. https://www.scientificamerican.com/article/the-imperfect-crime-how-the-wannacry-hackers-could-get-nabbed.

[6] Dmitry Ermilov, Maxim Panov, and Yury Yanovich. 2017. Automatic Bitcoin Address Clustering. In *Proceedings of 16th International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 461–466.

[7] Steven Goldfeder, Harry A. Kalodner, Dillon Reisman, and Arvind Narayanan. 2018. When the cookie meets the blockchain: Privacy risks of web payments via cryptocurrencies. *Proceedings on Privacy Enhancing Technologies* 2018, 4 (2018), 179–199.

[8] Yanan Gong, Kam-Pui Chow, Hing-Fung Ting, and Siu-Ming Yiu. 2022. Analyzing the Error Rates of Bitcoin Clustering Heuristics. In *Proceedings of the 2022 International Conference on Advances in Digital Forensics (IFIP) (IFIP Advances in Information and Communication Technology, Vol. 653)*. Springer, 187–205.

[9] Andy Greenberg. 2021. Prosecutors Trace $13.4M in Bitcoins From the Silk Road to Ulbricht's Laptop. https://www.wired.com/2015/01/prosecutors-trace-13-4-million-bitcoins-silk-road-ulbrichts-laptop.

[10] Rowel Gündlach, Martijn Gijsbers, David T. Koops, and Jacques Resing. 2021. Predicting confirmation times of Bitcoin transactions. *ACM SIGMETRICS Performance Evaluation Review* 48, 4 (2021), 16–19.

[11] David A. Harding and Peter Todd. 2015. Opt-in Full Replace-by-Fee Signaling. https://github.com/bitcoin/bips/blob/master/bip-0125.mediawiki.

[12] Martin Harrigan and Christoph Fretter. 2016. The Unreasonable Effectiveness of Address Clustering. In *Proceedings of the 2016 International Conference on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*. IEEE, 368–373.

[13] Danny Yuxing Huang, Maxwell Matthaios Aliapoulios, Vector Guo Li, Luca Invernizzi, Elie Bursztein, Kylie McRoberts, Jonathan Levin, Kirill Levchenko, Alex C. Snoeren, and Damon McCoy. 2018. Tracking Ransomware End-to-end. In *Proceedings of the 39th IEEE Symposium on Security and Privacy (S&P)*. IEEE, 618–631.

[14] Danny Yuxing Huang, Hitesh Dharmdasani, Sarah Meiklejohn, Vacha Dave, Chris Grier, Damon McCoy, Stefan Savage, Nicholas Weaver, Alex C. Snoeren, and Kirill Levchenko. 2014. Botcoin: Monetizing Stolen Cycles. In *Proceedings of the 21st Annual Network and Distributed System Security Symposium (NDSS)*. The Internet Society.

[15] Blockchain.com Inc. 2022. The world's most popular way to buy, sell, and trade crypto. https://www.blockchain.com/.

[16] Harshal Shridhar Kallurkar and B. R. Chandavarkar. 2022. Unconfirmed Transactions in Cryptocurrency: Reasons, Statistics, and Mitigation. In *Proceedings of 2022 International Conference on Public Key Infrastructure and its Applications (PKIA)*. IEEE, 1–7.

[17] Harry A. Kalodner, Malte Möser, Kevin Lee, Steven Goldfeder, Martin Plattner, Alishah Chator, and Arvind Narayanan. 2020. BlockSci: Design and applications of a blockchain analysis platform. In *Proceedings of the 29th USENIX Security Symposium (USENIX Security)*. USENIX Association, 2721–2738.

[18] George Kappos, Haaroon Yousaf, Rainer Stütz, Sofia Rollet, Bernhard Haslhofer, and Sarah Meiklejohn. 2022. How to Peel a Million: Validating and Expanding Bitcoin Clusters. In *Proceedings of the 31th USENIX Security Symposium (USENIX security)*. USENIX Association, 2207–2223.

[19] Dae-Yong Kim, Meryam Essaid, and Hongtaek Ju. 2020. Examining Bitcoin mempools Resemblance Using Jaccard Similarity Index. In *Proceedings of the 21st Asia-Pacific Network Operations and Management Symposium (APNOMS)*. IEEE, 287–290.

[20] Kyungchan Ko, Taeyeol Jeong, Sajan Maharjan, Chaehyeon Lee, and James Won-Ki Hong. 2019. Prediction of Bitcoin Transactions Included in the Next Block. In *Proceedings of the 1st International Conference on Blockchain and Trustworthy Systems (BlockSys) (Communications in Computer and Information Science, Vol. 1156)*. Springer, 591–597.

[21] Yang Li, Zilu Liu, and Zibin Zheng. 2019. Quantitative Analysis of Bitcoin Transferred in Bitcoin Exchange. In *Proceedings of the 1st International Conference on Blockchain and Trustworthy Systems (BlockSys) (Communications in Computer and Information Science, Vol. 1156)*. Springer, 549–562.

[22] Kevin Liao, Ziming Zhao, Adam Doupé, and Gail-Joon Ahn. 2016. Behind closed doors: measurement and analysis of CryptoLocker ransoms in Bitcoin. In *Proceedings of the 2016 APWG Symposium on Electronic Crime Research (eCrime)*. IEEE, 1–13.

[23] Feng Liu, Zhihan Li, Kun Jia, Panwei Xiang, Aimin Zhou, Jiayin Qi, and Zhibin Li. 2023. Bitcoin Address Clustering Based on Change Address Improvement. *IEEE Transactions on Computational Social Systems* (2023), 1–12.

[24] Eric Lombrozo, Johnson Lau, and Pieter Wuille. 2015. Segregated Witness (Consensus layer). https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki.

[25] Greg Maxwell. 2013. CoinJoin: Bitcoin privacy for the real world. https://bitcointalk.org/index.php?topic=279249.

[26] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. 2013. A fistful of bitcoins: characterizing payments among men with no names. In *Proceedings of the 2013 Internet Measurement Conference (IMC)*. ACM, 127–140.

[27] Malte Möser and Arvind Narayanan. 2022. Resurrecting Address Clustering in Bitcoin. In *Proceedings of the 26th International Conference on Financial Cryptography and Data Security (FC) (Lecture Notes in Computer Science, Vol. 13411)*. Springer, 386–403.

[28] Satoshi Nakamoto. 2009. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf

[29] Jonas David Nick. 2015. Data-Driven De-Anonymization in Bitcoin. In *Master's thesis*. ETH Zürich.

[30] United States Department of Justice. 2020. Global Disruption of Three Terror Finance Cyber-Enabled Campaigns. https://www.justice.gov/opa/pr/global-disruption-three-terror-finance-cyber-enabled-campaigns.

[31] OrdinalHub.com. 2023. Discover, Track & Analyze Bitcoin Ordinals. https://www.ordinalhub.com/.

[32] Masarah Paquet-Clouston, Bernhard Haslhofer, and Benoit Dupont. 2019. Ransomware payments in the Bitcoin ecosystem. *Journal of Cybersecurity* 5, 1 (2019), tyz003.

[33] Masarah Paquet-Clouston, Matteo Romiti, Bernhard Haslhofer, and Thomas Charvat. 2019. Spams meet Cryptocurrencies: Sextortion in the Bitcoin Ecosystem. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies (AFT)*. ACM, 76–88.

[34] Rebecca S. Portnoff, Danny Yuxing Huang, Periwinkle Doerfler, Sadia Afroz, and Damon McCoy. 2017. Backpage and Bitcoin: Uncovering Human Traffickers. In *Proceedings of the 23rd International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. ACM, 1595–1604.

[35] Fergal Reid and Martin Harrigan. 2011. An Analysis of Anonymity in the Bitcoin System. In *Proceedings of the 2nd International Conference on Privacy, Security, Risk and Trust (PASSAT)*. IEEE, 1318–1326.

[36] Casey Rodarmor. 2022. Ordinal Numbers. https://github.com/ordinals/ord/blob/master/bip.mediawiki.

[37] Muhammad Saad, Laurent Njilla, Charles A. Kamhoua, Joongheon Kim, DaeHun Nyang, and Aziz Mohaisen. 2019. Mempool optimization for Defending Against DDoS Attacks in PoW-based Blockchain Systems. In *Proceedings of the 2019 International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 285–292.

[38] Muhammad Saad, My T. Thai, and Aziz Mohaisen. 2018. POSTER: Deterring DDoS Attacks on Blockchain-based Cryptocurrencies through Mempool Optimization. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security (AsiaCCS)*. ACM, 809–811.

[39] Michele Spagnuolo, Federico Maggi, and Stefano Zanero. 2014. BitIodine: Extracting Intelligence from the Bitcoin Network. In *Proceedings of the 18th International Conference on Financial Cryptography and Data Security (FC) (Lecture Notes in Computer Science, Vol. 8437)*. Springer, 457–468.

[40] Ivo Stoepker, Rowel Gündlach, and Stella Kapodistria. 2021. Robustness analysis of Bitcoin confirmation times. *ACM SIGMETRICS Performance Evaluation Review* 48, 4 (2021), 20–23.

[41] Kai Wang, Jun Pang, Dingjie Chen, Yu Zhao, Dapeng Huang, Chen Chen, and Weili Han. 2022. A Large-scale Empirical Analysis of Ransomware Activities in Bitcoin. *ACM Transactions on the Web* 16, 2 (2022), 7:1–7:29.

[42] Qin Wang and Guangsheng Yu. 2023. Understanding BRC-20: Hope or Hype. (2023).

[43] Yunpeng Wang, Jin Yang, Tao Li, Fangdong Zhu, and Xiaojun Zhou. 2018. Anti-dust: A method for identifying and preventing Blockchain's dust attacks. In *Proceedings of the 2018 International Conference on Information Systems and Computer Aided Education (ICISCAE)*. IEEE, 274–280.

[44] Pieter Wuille, Jonas Nick, and Anthony Towns. 2020. Taproot: SegWit version 1 spending rules. https://github.com/bitcoin/bips/blob/master/bip-0341.mediawiki.

[45] Haaroon Yousaf, George Kappos, and Sarah Meiklejohn. 2019. Tracing Transactions Across Cryptocurrency Ledgers. In *Proceedings of the 28th USENIX Security*

Symposium (USENIX Security). USENIX Association, 837–850.
[46] Limeng Zhang, Rui Zhou, Qing Liu, Jiajie Xu, and Chengfei Liu. 2021. Transaction Confirmation Time Estimation in the Bitcoin Blockchain. In *Proceedings of the 22nd International Conference on Web Information Systems Engineering (WISE) (Lecture Notes in Computer Science, Vol. 13080)*. Springer, 30–45.
[47] Limeng Zhang, Rui Zhou, Qing Liu, Jiajie Xu, and Chengfei Liu. 2022. Bitcoin Transaction Confirmation Time Prediction: A Classification View. In *Proceedings of the 23rd International Conference on Web Information Systems Engineering (WISE) (Lecture Notes in Computer Science, Vol. 13724)*. Springer, 155–169.
[48] Baokun Zheng, Liehuang Zhu, Meng Shen, Xiaojiang Du, and Mohsen Guizani. 2020. Identifying the vulnerabilities of bitcoin anonymous mechanism based on address clustering. *Science China Information Sciences* 63, 3 (2020), 1–15.

APPENDIX

A IMPACT OF NODE COUNT ON COLLECTING UNCONFIRMED TRANSACTIONS

To further analyze the correlation between the number of unconfirmed transactions collected and the number of nodes, we deploy multiple nodes to collect unconfirmed transactions on December 3 and 4, 2023. Figure 8 demonstrates that the total number of unconfirmed transactions collected rarely increases once the number of nodes exceeds 5 in our environment.

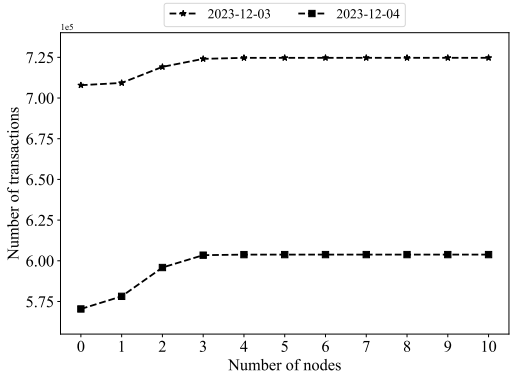

Figure 8: The number of transactions collected vs. the number of nodes running a modified Bitcoin core.

B FIELDS OF AN UNCONFIRMED TRANSACTION

Table 3 lists the fields of an unconfirmed transaction, showing many details that are not included in confirmed transactions.

C CASE STUDIES

Although unconfirmed transactions or even failed transactions are not stored in the blockchain, they can still reveal the motivations behind why users initiate these transactions. This section demonstrates the high value of failed transactions in analyzing transaction behaviors through two case studies.

C.1 Analysis of Binance Exchange Address

One of the primary safeguards employed by Bitcoin exchanges to prevent attacks is the utilization of cold and hot storage technology, accompanied by carefully designed risk control systems [21].

Table 3: Fields of an unconfirmed transaction. Fields with a star (*) are only present in unconfirmed transactions.

Name	Content
txid	hash of transaction, not including witness data.
wtxid	hash of transaction, including witness data.
inputs	the inputs of transaction.
outputs	the outputs of transaction.
fee	transaction fee in BTC
vsize	virtual transaction size
weight	transaction weight
time*	local time when the transaction enters mempool
removetime*	local time when the transaction is removed
height*	block height when transaction enters mempool
descendantcount*	number of descendant transactions
descendantsize*	vsize of descendant transactions
descendantfees*	modified fees of descendant transactions
ancestorcount*	number of ancestor transactions
ancestorsize*	vsize of ancestor transactions
ancestorfees*	modified fees of ancestor transactions
depends*	unconfirmed transactions used as inputs
spentby*	unconfirmed transactions spending outputs
replaceable*	whether this transaction could be replaced

There are three main types of addresses controlled by Bitcoin exchanges: hot wallet addresses, cold wallet addresses, and user wallet addresses. The primary function of the hot wallet is to maintain a pool of bitcoins for user withdrawal demand. The user wallet address is created by the exchange, and the exchange holds the private key of the address. Users can deposit bitcoins to the exchange using the user wallet address.

Figure 9 shows a failed Binance transaction, its replacement confirmed transaction, and two other confirmed transactions related to Binance exchange. The address *bc1q...7s3h* is labeled with Binance exchange by Blockchain.com [15]. When the transaction *66c1...9dc6* is confirmed, the transaction *dc43...ef51* is initiated to transfer bitcoins in the address *bc1q...wwvq* to Binance exchange address. Three minutes after initiating the transaction *dc43...ef51*, the same user initiates the transaction *4161... bb25*. The second transaction spends the same UTXO but pays a much higher fee. As a result, miners choose to store the second transaction in a block, which is eventually confirmed. The transaction *dc43...ef51* turns out to be a failed transaction, which is removed by all Bitcoin nodes and can never be confirmed again. Then another transaction *6903...4583* with 100 inputs and 1 output is initiated, transferring bitcoins from the address *19Fa...Hd5X* to the address *bc1q...7s3h*.

From the content of the transaction *dc43...ef51* and the transaction *6903...4583*, the purpose of the address *bc1q...wwvq* is to transfer its bitcoins to the address *bc1q...7s3h*. A question worth analyzing is why the user would replace the transaction *dc43...ef51* with the second one *4161...bb25*.

To answer this question, we investigate 127 confirmed transactions involving the address *bc1q...wwvq*. All outgoing transactions of the address *bc1q...wwvq* consist of one input and one output. When the address *bc1q...wwvq* transfers bitcoins to other addresses, the recipient address is always the address *19Fa...Hd5X*. The address *19Fa...Hd5X* then transfers bitcoins to the address *bc1q...7s3h*. Both incoming and outgoing transactions of the address *bc1q...wwvq* occur in pairs, i.e., the address receives bitcoins and then transfers out all the bitcoins it receives. Therefore, we infer that the transaction

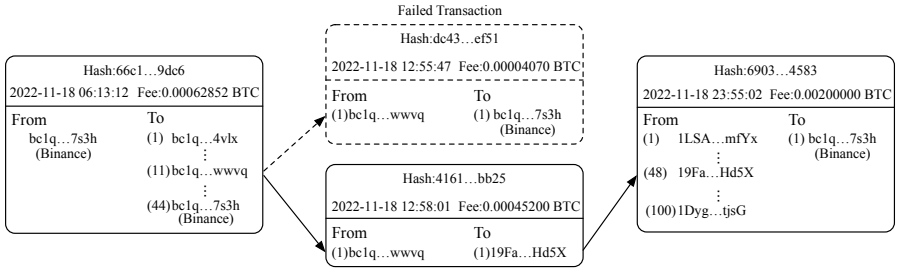

Figure 9: Analysis of Binance Exchange Address.

dc43...ef5 originates from an error made by Binance exchange. The address *bc1q...7s3h* is a hot wallet address, and under the rules of Binance exchange, the address *bc1q...wwvq* cannot transfer bitcoins directly to the address *bc1q...7s3h*. Therefore, when the transaction *dc43...ef51* is initiated, the operator or script discovers the misoperation and initiates another high-fee transaction *4161...bb25* to replace the first one.

The analysis indicates that Binance exchange manages a variety of addresses, each with its own unique role and set of duties. Binance exchange coordinates the interactions between these addresses carefully to prevent unauthorized Bitcoin transactions.

By analyzing unconfirmed transactions, we can shed light on the internal risk prevention and control mechanisms of Binance exchange. This information can assist regulators in verifying the cryptocurrency exchange's reported information and inadvertently disclosed transfer behavior.

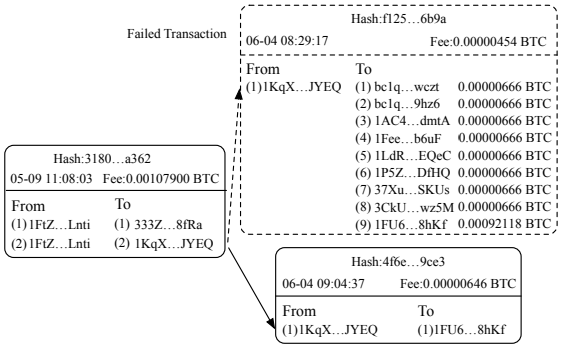

Figure 10: Dust attack against whale addresses.

C.2 Dust Attacks Against Whale Addresses

The dust attack is defined as malicious behavior that targets Bitcoin users and privacy by sending tiny amounts of bitcoins to victims' addresses [43]. The dust attacker attempts to reveal the user's identity by collecting data on the aggregation points of tiny amounts when the user initiates a new transaction through his cryptocurrency wallet software. The attackers track the transaction activity of these addresses in an attempt to link the dusted addresses and identify the entity behind them [43].

Figure 10 shows a potential dust attack against whale addresses found in a failed transaction. Figure 10 contains two confirmed transactions and one failed transaction. In this scenario, the transaction *3180...a362* with two inputs and two outputs is initiated first. When this transaction is confirmed, the address *1KqX...JYEQ* transfers its received bitcoins to the address *1FU6...8hKf* through the transaction *4f6e...9ce3*. We cannot find anything unusual about this address *1KqX...JYEQ* from confirmed transactions.

However, before the transaction *4f6e...9ce3* is initiated, the transaction *f125...6b9a* is initiated. These two transactions spend the same UTXO *1KqX...JYEQ*. Due to the much higher fee of the transaction *4f6e...9ce3*, miners choose to store the transaction *4f6e...9ce3* in a block that is eventually confirmed. Therefore, the transaction *f125...6b9a* turns out to be a failed transaction. However, the failed transaction *f125...6b9a* reflects the user's malicious behavior, which is not reflected in confirmed transactions.

More specifically, the transaction *f125...6b9a* has one input and 9 outputs. Note that all 8 outputs of this transaction have the same revenue, i.e. 0.00000666 BTC ($ 0.12). The remaining output is the change address of the sender. The addresses *bc1q...wczt*, *bc1q...9hz6*, and *1Fee...b6uF* are labeled by Blockchain.com [15] as *FBI3*, *FBI2 (Silk Road)* and *MtGox Hacker* respectively. Stolen bitcoins from the Bitfinex Hack continue to converge to the address *bc1q...wczt*, which currently has a balance of over 94,643 BTC without any outgoing transfers. Nearly 70,000 BTC confiscated by the U.S. government from the black market site *Silk Road* are transferred to the address *bc1q...9hz6* in early November, 2020. To date, the bitcoins have not been moved or liquidated. Former Mt.Gox CEO Mark Karpeles confirms that the bitcoins residing at the address *1Fee...b6uF* are stolen from the Mt.Gox exchange. The address *1P5Z...DfHQ* is controlled by FTX exchange, which declares bankruptcy on November 14, 2022. The other addresses except *1FU6...8hKf* also have a large balance when this failed transaction *f125...6b9a* is initiated. Currently, the balance of the address *3CkU...wz5M* is 0. However, the balance of the address *3CkU...wz5* is more than 50,620 BTC when the failed transaction *f125...6b9a* is initiated. Therefore, the address *1KqX...JYEQ* initiates transaction *f125...6b9a*, sending a tiny amount of bitcoins to multiple whale addresses in an attempt to conduct a dust attack.

Focusing only on confirmed transactions in the blockchain may not reveal the true intentions of users. Unconfirmed transactions, however, can be used to identify malicious behavior that may not be identified by confirmed transactions alone. Additionally, unconfirmed transactions provide an opportunity to identify and understand malicious behavior before it occurs.

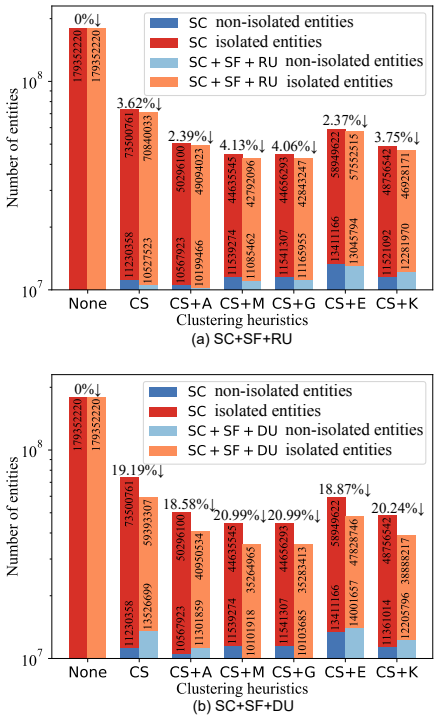

Figure 11: The individual contributions of our proposed heuristics.

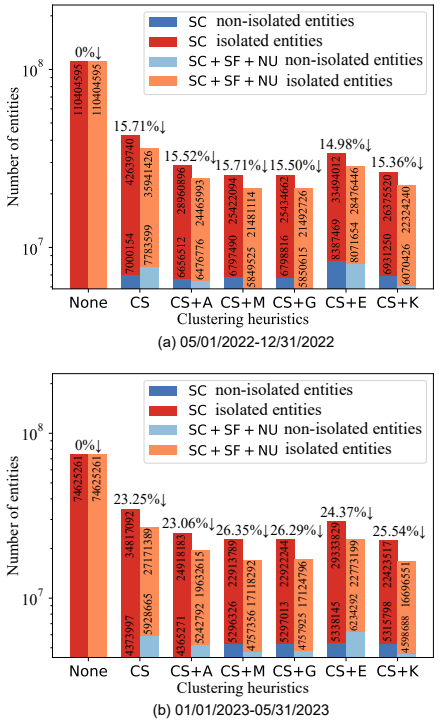

Figure 12: Comparison of the number of entities in the SC clustering result and the SC+SF+NU clustering results in different periods.

D DETAILED RATIONALE FOR LABELING EFFECTIVENESS

Since taproot script can only be created from existing taproot outputs [44], inscriptions are created using a two-phase (commit/reveal) procedure. In the first phase, a taproot output is created in the commit transaction, committing to a script that contains the inscription content. In the second phase, the reveal transaction spends the output of the commit transaction, thereby revealing the inscription content in the blockchain.

In the early stages, the provenance of inscriptions is only traceable through off-chain data and social consensus, such as the website [31]. The introduction of the parent-child inscription mechanism transforms the tracing process. This mechanism involves using the existing inscription as an input in the reveal transaction, thereby designating it as the parent of the new inscription. This mechanism effectively demonstrates that the creator of the child inscription controls the parent inscription.

E THE INDIVIDUAL CONTRIBUTIONS OF OUR PROPOSED HEURISTICS

To further analyze the effect of our proposed clustering heuristics, we conduct additional experiments to quantify the individual contributions of the two proposed clustering heuristics to the improvement of the clustering results.

To describe experimental results clearly, we assign a name to each clustering result, composed of the heuristic and the state of transactions. The RU clustering result refers to the result of applying the *replacement change* heuristic to unconfirmed transactions. The DU clustering result refers to the result of applying the *dependency chain* heuristic to unconfirmed transactions.

Figure 11 demonstrates that the *dependency chain* heuristic contributes significantly to the improvement. This result is related to the large proportion of transactions that form the dependency chain, described in Section 4.1.

F FURTHER TEMPORAL ANALYSIS

To further analyze the impact of our proposed heuristics before and after the emergence of ordinal inscriptions, we conduct two experiments, one before January 2023 and one after, to highlight significant variations. Figure 12 shows that beginning in January 2023, with the emergence of ordinal inscriptions, the impact of our proposed heuristic has become more pronounced.