2021 International Conference on Data Mining Workshops (ICDMW) | 978-1-6654-2427-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICDMW53433.2021.00102

Identifying Darknet Vendor Wallets by Matching Feedback Reviews with Bitcoin Transactions

1st Xucan Chen
Computer Science
Georgia State University
Atlanta, USA
xchen41@student.gsu.edu

2nd Wei Cheng
NEC Laboratories America, Inc
Princeton, USA
weicheng@nec-labs.com

3th Marie Ouellet
Criminal Justice and Criminology
Georgia State University
Atlanta, USA
mouellet@gsu.edu

4rd Yuan Li
Computer Science
North China University of Technology
Beijing, China
neu_liyuan@126.com

5th David Maimon
Criminal Justice and Criminology
Georgia State University
Atlanta, USA
Dmaimon@gsu.edu

6th Yubao Wu
Computer Science
Georgia State University
Atlanta, USA
ywu28@gsu.edu

***Abstract*—Darknet markets are e-commerce websites operating on the darknet and have grown rapidly in recent years. Darknet only allow cryptocurrencies as the payment methods, making it hard for law enforcement to trace those illicit transactions. In this paper, we present a method to identify vendors' bitcoin addresses by matching vendors' feedback reviews with bitcoin transactions in the public ledger. The problem is decomposed into two steps in formulation. In Step 1, we solve a bounding box matching between the set of feedback reviews and bitcoin transactions. In Step 2, we find the bitcoin addresses with a maximum coverage of the reviews. Baseline algorithm for Step 1 runs in quadratic time thus we develop a K-D tree to accelerate the computing. Problem in Step 2 is NP-hard thus we develop a greedy algorithm with an approximation ratio of $(1 - 1/e)$ based on the submodular property of the objective function. We further propose a cost-effective algorithm to accelerate both steps effectively. Comprehensive experimental results have demonstrated the effectiveness and efficiency of the proposed method.**

***Index Terms*—bitcoin, darknet, submodular**

I. Introduction

Illegal online sales have grown exponentially [1]. The privacy of participants in illicit online transactions is protected through both the darknet and cryptocurrency. The darknet utilizes The Onion Router (Tor) network to hide users' IP addresses from the internet service provider. A darknet market is a commercial website operating on the darknet. Users can surf those web pages through the Tor browser without worrying about the leakage of their IP addresses. Even users are anonymous, darknet markets publish vendors' feedback reviews so that buyers can evaluate vendors' reputation. Darknet markets choose cryptocurrencies as payment currency mainly because they are also anonymous. Both buyers and vendors can trade anonymously through cryptocurrencies [2]. Unlike traditional currencies, cryptocurrencies like Bitcoin [3] are decentralized: there is no central authority responsible for the issuance of cryptocurrencies and there is no need to involve a trusted third-party like banks when making online transfers [4], [5].

We focus on Bitcoin in this paper because Bitcoin is the most popular cryptocurrency which is accepted by all darknet markets [6]. Using blockchain and distributed ledger technology, the Bitcoin system promises great transparency and improved trust across transaction value chains [7], [8]. Without a third-party to ensure transaction, the Bitcoin system publishes all of its history transaction data. The Bitcoin ledger stores all transaction records in history which is public to any Bitcoin users. A user wallet can own multiple bitcoin addresses, which are the "pseudonymous identity" of this user in the public ledger.

This paper aims at finding vendors' Bitcoin addresses used in the darknet markets by matching feedback reviews with bitcoin transactions. To narrow down the scope of the problem, we choose Bitcoin and Wall Street Market as a study example. Each feedback review is matched to a bitcoin transaction based on timestamp and value transferred in this transaction. Therefore a Bitcoin address whose history transactions can match more reviews of a vendor have a higher possibility to belong to this vendor. Specifically, we decompose our problem formulation into two sub-problems: Bounding Box Matching Problem and Maximum Review Coverage Problem. In the Bounding Box Matching Problem, we construct a bounding box for each review and find matched Bitcoin transactions. We build a K-D tree from massive Bitcoin transaction data to achieve quick range searching in a bounding box. In the Maximum Review Coverage Problem, we prove the NP-Hardness of the problem. We exploit the submodular property of the objective function and design a greedy algorithm with an approximation ratio of $(1 - 1/e)$ to find a set of addresses that can cover near-optimal product reviews received by one vendor. Our method can discover the number of addresses used by one vendor, realizing one-to-many mapping. We further develop an algorithm that can effectively accelerate

978-1-6654-2427-1/21/$31.00 ©2021 IEEE
DOI 10.1109/ICDMW53433.2021.00102

the matching and greedy algorithm.

Our contributions are as follows:

- We propose the problem of identifying the vendors' Bitcoin addresses by matching public Bitcoin transactions to vendor's feedback reviews in darknet markets. This problem is important because of two potential applications. First, it helps law enforcement to trace illicit transactions. Second, it reveals a privacy concern of cryptocurrencies so helps better design new cryptocurrencies.
- We decompose the complicated problem into two sub-problems and provide efficient computing algorithms for the sub-problems. We further propose Cost-Effective Addresses Searching(CEAS) algorithm to accelerate the whole process, which can reduce about 60% matching calculations in experiments.
- We extensively evaluate our methods in both real and synthetic data and demonstrate the effectiveness and accuracy of our method.

The remainder of this paper is organized as follows. Section 2 introduces details about Bitcoin transactions behind different markets and reviews related work, followed by the formulation of the problem in Section 3. Section 4 proposes our algorithm and related computational complexity analysis. Section 5 details our experiments and section 6 concludes the paper.

II. Background

In this section, we will introduce Bitcoin transaction mechanisms behind different darknet sites and related works.

A. Bitcoin Transactions Behind Darknet Markets

Here we describe our experiments of purchases in Wall-Street market, one of the most popular darknet markets in the last few years. Since we know the start point (buyer address) of the transaction, we can track Bitcoin flows from the start point during our purchase.

In darknet markets, buyers do not send Bitcoin to vendors directly. All darknet markets provide escrow services to avoid scams and protect both buyers and vendors. In each market, two operations are performed by the buyer: order and confirm the order. Like normal online shopping, buyers need to order products first and they can confirm this order once they receive the product. Research shows whether we can observe related Bitcoin transactions in public ledger for these two operations in some biggest darknet markets [9]. In the Wall Street Market, when we order a product or confirm this order, a related bitcoin transaction occurs .

Figure 1 shows the resulting Bitcoin flow during these two operations by a buyer. A consumer needs to send Bitcoin to a newly generated escrow address after he places an order. Bitcoin will stay in this escrow address until the buyer confirms the order when they received the product. The confirmation will trigger the Bitcoin transfer from the escrow address to the vendor's address through a mixed transaction. Mixed transactions are utilized to break the direct connection between the sender and receiver address by combining several transactions into one transaction with multiple senders and

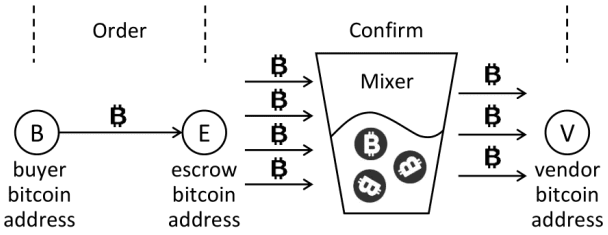


Fig. 1. Bitcoin flow of one purchase in Wall Street Market

multiple receivers [10], [11].However, we still can get the amount of Bitcoin received by each receiver address in mixed transaction. In Wall Street Market, 94.5% of Bitcoin in escrow address is sent to the vendor's address, and the remaining 5.5% is transferred to the Wall Street market address as a commission fee during a mixed transaction.

When the buyer confirms that they have received the illicit product, they need to write feedback that appears in the review list of products with time and amount of dollar spent during this transaction. If we could find the related Bitcoin transaction in the public ledger, the receiver of this transaction is the Bitcoin address from the vendor.

Now we know that in the Wall Street market a confirmation of a purchase will result in a Bitcoin transaction from the escrow address to the vendor's address. Experiments we have show this transaction occurs within an hour after confirmation from the buyer. The darknet market will take some time to mixed this transaction to other Bitcoin transactions from outside the darknet. Therefore we can tell the time review posted is close to the time when a mixed transaction happened.

A review of the product is required immediately from buyer after the confirmation. In each review, we can gather information including money spent and time when this review post. Experiments we have in Wall Street Market show the time when the review posted is close to the time when a mixed transaction happened and the price of a product should be close to the bitcoin value received by the vendor's address.

B. Related Work

1) Bitcoin De-anonymization: The rise of Bitcoin has increased researchers' interest in privacy provided in crytocurrency [12]–[16], also the usage of Bitcoin in darknet [17]–[19]. To attack the privacy of Bitcoin, the most common way is to study the Bitcoin transaction graph after clustering the Bitcoin address from one wallet. A wallet represents an entity. A user stores all addresses in a wallet. Researchers have widely clustered Bitcoin addresses heuristically. Androulaki tested the effectiveness of the Bitcoin address clustering methods with stimulation [14]. Spagnuolo link the clustered addresses to the Silk Road escrow address exposed by FBI and analyze the Bitcoin flow related to this escrow address [12]. Fleder not only linked the clustered addresses to Silk Road escrow but also link Bitcoin addresses to some public entities [13]. PageRank is then applied on the transaction graph to find addresses that are close to the Silk Road escrow address.

TABLE I
MAIN SYMBOLS

symbols	definitions
$b(t,u,r)$	a bitcoin transaction with timestamp t, money value u, and receiving address r
B	a set of bitcoin transactions, $B = \{b_i(t_i, u_i, r_i)\}$
R	a set of receiving bitcoin addresses, $R = \{r_i\}$
$f(\tau, v)$	a feedback review with timestamp t and money value v
F	a set of feedback reviews, $F = \{f_j(\tau_j, v_j)\}$, from one vendor
θ, ϕ	bounding box thresholds of timestamp and money value
$S(R)$	a set function, input R: a set of receiving bitcoin addresses, output: the set of feedback reviews matched with R
$s(R)$	a set function, input R: a set of receiving bitcoin addresses, output: the number of feedback reviews matched with R, i.e., $s(R) = \|S(R)\|$, where $\|\cdot\|$ represents the cardinality of a set

In the transaction graph, each node represents an entity that contains all addresses owned by this entity. However, current clustering methods don't work for mixed transactions that combine several transactions into one transaction with multiple senders and multiple receivers. Mix is quite common in Bitcoin transactions, especially illegal transactions in recent years. In this paper, our method is the first one to explore relationships between feedback of vendors and receiver parts of Bitcoin transactions which won't be affected by mixed transactions.

2) Matching: If we treat each review as one type of node and Bitcoin transaction as another type of node, reviews of a vendor and transactions from an address can form a bipartite graph after we link the review to the matched transaction. Portnoff matches specific ads to publicly available Bitcoin transactions based on the cost of ads and timestamp at which the ad was placed [20]. Hungarian algorithm and Hopcroft-Karp algorithm are two greedy algorithm can successfully find the maximum matching in bipartite graph [21]–[23]. We can use these algorithms to find the maximum matching between a vendor and a Bitcoin address. However, this method is not effective enough to match multiple addresses. A vendor in the darknet normally owns a lot of bitcoin addresses.

3) Submodular Function: Submodular optimization algorithm has been exploited in other areas before [24]–[27]. Kempe using a greedy algorithm based on submodular function to maximize the influence in social network [28]. Leskovec exploits the submodularity of outbreak detection to develop an efficient approximation algorithm for water distribution and the blogosphere monitoring problem [29].

III. PROBLEM FORMULATION

In this section, we formulate the problem of matching bitcoin transactions with feedback reviews. Table I shows the main symbols used in this paper and their definitions.

Let $B = (b_1(t_1, u_1, r_1), b_2(t_2, u_2, r_2), \cdots, b_n(t_n, u_n, r_n))$ represents the set of all bitcoin transactions, where $b_i(t_i, u_i, r_i)$ represents a bitcoin transaction with three attributes: timestamp t_i, money value u_i, and receiving address r_i. Here we only need receiver part of these bitcoin transactions. Let R represent a set of all unique receiving addresses in B. Let $F = (f_1(\tau_1, v_1), f_2(\tau_2, v_2), \cdots, f_m(\tau_m, v_m))$ represents a list of feedback reviews received by one vendor, where $f_j(\tau_j, v_j)$ is a feedback review with two attributes: timestamp τ_j and money value v_j.

TABLE II
AN EXAMPLE TO ILLUSTRATE THE VENDOR RECEIVING ADDRESS SET PROBLEM

Input		After matching		
all bitcoin transactions B	feedback reviews F	feedback reviews F	bitcoin transactions	receiving addresses
$b_1(t_1, u_1, r_2)$	$f_1(\tau_1, v_1)$	$f_1(\tau_1, v_1)$	$b_1(t_1, u_1, r_2)$	r_2
$b_2(t_2, u_2, r_3)$	$f_2(\tau_2, v_2)$		$b_2(t_2, u_2, r_3)$	r_3
$b_3(t_3, u_3, r_4)$	$f_3(\tau_3, v_3)$		$b_3(t_3, u_3, r_4)$	r_4
$b_4(t_4, u_4, r_5)$	$f_4(\tau_4, v_4)$		$b_4(t_4, u_4, r_5)$	r_5
$b_5(t_5, u_5, r_6)$	$f_5(\tau_5, v_5)$		$b_5(t_5, u_5, r_6)$	r_6
$b_6(t_6, u_6, r_1)$	$f_6(\tau_6, v_6)$	$f_2(\tau_2, v_2)$	$b_6(t_6, u_6, r_1)$	r_1
$b_7(t_7, u_7, r_7)$	$f_7(\tau_7, v_7)$		$b_7(t_7, u_7, r_7)$	r_7
$b_8(t_8, u_8, r_1)$		$f_3(\tau_3, v_3)$	$b_8(t_8, u_8, r_1)$	r_1
$b_9(t_9, u_9, r_8)$			$b_9(t_9, u_9, r_8)$	r_8
$b_{10}(t_{10}, u_{10}, r_1)$		$f_4(\tau_4, v_4)$	$b_{10}(t_{10}, u_{10}, r_1)$	r_1
$b_{11}(t_{11}, u_{11}, r_2)$			$b_{11}(t_{11}, u_{11}, r_2)$	r_2
$b_{12}(t_{12}, u_{12}, r_3)$			$b_{12}(t_{12}, u_{12}, r_3)$	r_3
$b_{13}(t_{13}, u_{13}, r_4)$			$b_{13}(t_{13}, u_{13}, r_4)$	r_4
$b_{14}(t_{14}, u_{14}, r_9)$			$b_{14}(t_{14}, u_{14}, r_9)$	r_9
$b_{15}(t_{15}, u_{15}, r_1)$		$f_5(\tau_5, v_5)$	$b_{15}(t_{15}, u_{15}, r_1)$	r_1
$b_{16}(t_{16}, u_{16}, r_3)$			$b_{16}(t_{16}, u_{16}, r_3)$	r_3
$b_{17}(t_{17}, u_{17}, r_5)$			$b_{17}(t_{17}, u_{17}, r_5)$	r_5
$b_{18}(t_{18}, u_{18}, r_1)$		$f_6(\tau_6, v_6)$	$b_{18}(t_{18}, u_{18}, r_1)$	r_1
$b_{19}(t_{19}, u_{19}, r_3)$			$b_{19}(t_{19}, u_{19}, r_3)$	r_3
$b_{20}(t_{20}, u_{20}, r_2)$		$f_7(\tau_7, v_7)$	$b_{20}(t_{20}, u_{20}, r_2)$	r_2
$b_{21}(t_{21}, u_{21}, r_{10})$			$b_{21}(t_{21}, u_{21}, r_{10})$	r_{10}
...				

Problem 1: **Vendor Receiving Address Set Problem**:Finding a set of receiving addresses $R_k \subset R$ which are likely the bitcoin addresses in the vendor's wallet, according to the matching between the vendor's feedback reviews in set F and bitcoin transactions in set B.

Based on practical observations, the timestamp τ_j and money value v_j in $f_j(\tau_j, v_j)$ are approximately equal to the timestamp t_i and money value u_i in a bitcoin transaction $b_i(t_i, u_i, r_i)$ respectively, i.e., $\tau_j \approx t_i$ and $u_j \approx v_i$, if $b_i(t_i, u_i, r_i)$ is the corresponding bitcoin transaction of the feedback review $f_j(\tau_j, v_j)$. By comparing the timestamp and money value attributes, we can match feedback reviews to bitcoin transactions thus find the receiving addresses. In case the vendor does not change the receiving addresses frequently, many of their feedback reviews will be matched with bitcoin transactions with the same receiving addresses. Problem 1 aims at finding a set of receiving addresses $R_k \subset R$ whose involved bitcoin transactions match the maximum number of feedback reviews in F.

We use a bounding box to find candidate bitcoin transactions for a feedback review. For each review, we search the bitcoin transactions B with a bounding box. We set thresholds θ and ϕ to constrict the ranges of timestamp and money value respectively. Currently, there are thousands of bitcoin transactions every second. We compare a feedback review with a bitcoin transaction and we are able to match a review to thousands of candidate transactions in real data with the range we provided. According to research [9], a bitcoin transaction happens within

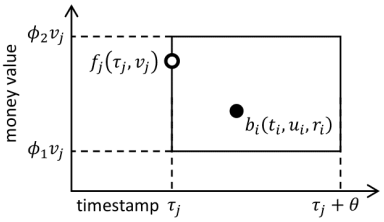


Fig. 2. Bounding boxes of timestamp and money value for matching a feedback review with a bitcoin transaction

an hour after a buyer posts the review with a high possibility. Therefore, we use $\tau_j \leq t_i \leq \tau_j + \theta$ as the bounding box for timestamp and set $\theta = 1$ hour. The market takes 5.5% commission fee and the exchange rate also fluctuates in a day. Therefore, we use $\phi_1 v_j \leq u_i \leq \phi_2 v_j$ as the bounding box for money value. If the fluctuation range is 10%, the setting will be $\phi_1 = (1 - 5.5\%)(1 - 10\%) = 0.8505$ and $\phi_2 = (1 - 5.5\%)(1 + 10\%) = 1.0395$. Figure 2 illustrates the bounding boxes. If a bitcoin transaction $b_i(t_i, u_i, r_i)$ falls in the bounding boxes of a feedback review $f_j(\tau_j, v_j)$, we say b_i and f_j are a match, which is formally defined in Definition 1.

Definition 1: f_j **is matched with** b_i : A feedback review $f_j(\tau_j, v_j)$ is matched with a bitcoin transaction $b_i(t_i, u_i, r_i)$ if $\tau_j \leq t_i \leq \tau_j + \theta$ and $\phi_1 v_j \leq u_i \leq \phi_2 v_j$.

Definition 2: f_j **is matched with** r_i : A receiving bitcoin address r_i is matched with a feedback review $f_j(\tau_j, v_j)$ if $f_j(\tau_j, v_j)$ is matched with a bitcoin transaction $b_i(t_i, u_i, r_i)$ where r_i is the receiving address.

Table II shows an example. The left part of Table II shows the input set B of bitcoin transaction and the input set F of feedback reviews received by a vendor. Because of the limited space, only 21 bitcoin transactions are shown here but B should contain hundreds of millions of transactions. In the example, the vendor has 7 feedback reviews. The right part of Table II shows the matches between the 7 feedback reviews and the 21 bitcoin transactions. To save space, we ignore the concrete values of timestamps and money values in B and F in the left part. Whenever there is a match, it means that the feedback review and bitcoin transaction satisfy Definition 1. From the right part of II, we can see that there could be multiple bitcoin transactions that are matches of one feedback review. In real cases, a feedback review could match hundreds of bitcoin transactions. And the receiving addresses in those matched bitcoin transactions are the candidate addresses of the vendor.

In order to accelerate the computation in the later stages, we transfer the matching results to the vertical and binary formats in Table III. To solve Problem 1, we aim at finding a minimum set of bitcoin addresses covering all feedback reviews. The intuition is that a bitcoin address that can cover many feedback reviews is very likely to belong to the vendor wallet. This is especially true when the careless vendor infrequently or barely changes their bitcoin address.

Let $S(R)$ represents a set function which returns the set

TABLE III
VERTICAL AND BINARY FORMAT

receiving bitcoin address r_i	feedback reviews $S(\{r_i\})$ matched with r_i	support $s(\{r_i\})$	binary format representation
r_1	f_2, f_3, f_4, f_5, f_6	5	0, 1, 1, 1, 1, 1, 0
r_2	f_1, f_4, f_7	3	1, 0, 0, 1, 0, 0, 1
r_3	f_1, f_4, f_5, f_6	4	1, 0, 0, 1, 1, 1, 0
r_4	f_1, f_4	2	1, 0, 0, 1, 0, 0, 0
r_5	f_1, f_5	1	1, 0, 0, 0, 1, 0, 0
r_6	f_1	1	1, 0, 0, 0, 0, 0, 0
r_7	f_2	1	0, 1, 0, 0, 0, 0, 0
r_8	f_3	1	0, 0, 1, 0, 0, 0, 0
r_9	f_4	1	0, 0, 0, 1, 0, 0, 0
r_{10}	f_7	1	0, 0, 0, 0, 0, 0, 1

of feedback reviews that are matched with the input set R of bitcoin addresses. Let $s(R)$ represent a set function which returns the number of feedback reviews matched with the input set R, i.e., $s(R) = |S(R)|$. Theorem 1 shows that $s(R)$ is a submodular set function [28].

Theorem 1: Given three bitcoin address sets A, B, C with $A \subseteq B \subseteq C$ and a bitcoin address $r \in C \setminus B$, we have

$$s(A \cup \{r\}) - s(A) \geq s(B \cup \{r\}) - s(B)$$

The left equation $s(A \cup \{r\}) - s(A)$ represents the number of feedback reviews that are newly matched after adding the bitcoin address r to the set A. Thus, we have $s(A \cup \{r\}) - s(A) = s(\{r\}) - |S(A) \cap S(\{r\})|$. Similarly, $s(B \cup \{r\}) - s(B) = s(\{r\}) - |S(B) \cap S(\{r\})|$. Since $A \subseteq B$, A are matched with less or equal feedback reviews than B, i.e., $S(A) \subseteq S(B)$, we have $S(A) \cap S(\{r\}) \subseteq S(B) \cap S(\{r\})$ thus $|S(A) \cap S(\{r\})| \leq |S(B) \cap S(\{r\})|$. Therefore, $s(\{r\}) - |S(A) \cap S(\{r\})| \geq s(\{r\}) - |S(B) \cap S(\{r\})|$. This completes the proof.

Theorem 1 exhibits the diminishing returns property, which is the equivalent condition of a submodular set function. The property can be explained as that the marginal gain from adding a bitcoin address to the set R is at least as high as the marginal gain from adding the bitcoin address to a superset of R. We aim at finding a receiving address set R with size k, i.e., $|R| = k$, which can maximize $s(R)$.

We decompose Problem 1 into two steps, which are formulated as Problem 2 and Problem 3. Problem 2 aims at matching bitcoin transactions with feedback reviews. Problem 3 aims at searching the optimal set of bitcoin addresses for a vendor. The output of Problem 2 is the input of Problem 3.

Problem 2: **Bounding Box Matching Problem**: Given the set of bitcoin transactions B, the set of feedback reviews F, and the bounding box parameters θ for timestamp, ϕ_1 and ϕ_2 for money values, the problem aims at finding a family of sets $\{S_i\}$, where S_i represents the set of feedback reviews covered by candidate receiving address r_i.

Problem 3: **Maximum Review Coverage Problem**: Given a family of sets $\{S_i\}$ with S_i representing the set of feedback reviews matched with each r_i in $b_i(t_i, u_i, r_i) \in B$ and a positive integer k as the budget for the number of receiving addresses, the problem is finding an address set R with size

Algorithm 1: BasicBoundingBoxSearch(B, F, θ, ϕ_1, ϕ_2)

Input: bitcoin transaction set B, feedback review set F, bounding box parameter θ for timestamp, bounding box parameters ϕ_1 and ϕ_2 for money value
Output: a family of sets $\{S_i\}$, where S_i represents the set of feedback reviews matched with r_i in $b_i(t_i, u_i, r_i) \in B$

for each unique r_i in $b_i(t_i, u_i, r_i) \in B$ **do** $S_i \leftarrow \emptyset$; // initialization
$f_j(\tau_j, v_j) \in F : b_i(t_i, u_i, r_i) \in B$: **if** $\tau_j \leq t_i \leq \tau_j + \theta$ **and** $\phi_1 v_j \leq u_i \leq \phi_2 v_j$ **then**
  $S_i \leftarrow S_i \cup \{f_j\}$;

k, i.e., $|R| = k$, that are matched with the maximum number of feedback reviews. That is, finding the optimal solution R of the following optimization problem:

$$\begin{aligned} \max \quad & s(R) \\ \text{s.t.} \quad & |R| = k \end{aligned}$$

Problem 2 can be solved in polynomial time.

Theorem 2: Problem 3 is NP-hard.

Problem 3 can be reduced from the famous Set Cover problem [30]. Let $X = \{X_1, \cdots, X_q\}$ be a family of sets with $Y = \{y_1, \cdots, y_p\} = \bigcup_{i=1}^{q} X_i$ being the elements. The NP-complete Set Cover problem aims at finding whether there exist k of the subsets in $\{X_i\}$ whose union is equal to Y. Given an arbitrary instance of the Set Cover problem, we define a corresponding instance of Problem 3. For each subset $X_i \in X$, we create a bitcoin address r_i and a set S_i. Therefore, we get a family of sets $\{S_i\}$ and $R_{\text{all}} = \{r_1, \cdots, r_q\}$, where all r_i's are unique. For each element y_j, we create a feedback review f_j. Therefore, we get $F = \{f_1, \cdots, f_p\}$. We add f_j to S_i, i.e., r_i is matched with f_j, if and only if X_i contains y_j. The Set Cover problem is equivalent to deciding if there is a set $R \subseteq R_{\text{all}}$ of k bitcoin addresses with $s(R) = p$.

IV. Computing Algorithms

In this section, we discuss how to efficiently compute the problems. We first study the bounding box and KD tree techniques for matching the bitcoin transactions with feedback reviews. We then exploit a greedy algorithm that can obtain an address set that is provably cover $(1-1/e)$ ratio reviews of optimal. Here e is Euler's number. Finally, we propose our fast method which can accelerate the whole process by effectively reducing the times of matching.

A. Bounding Box and K-D tree

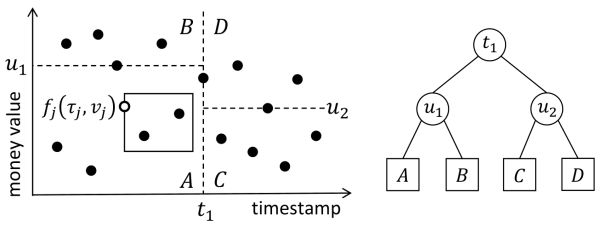
For each pair of feedback review $f \in F$ and bitcoin transaction $b \in B$, we need to check the inequalities of timestamp and bitcoin value. Let $n = |B|$ be the number of bitcoin transactions in B and $m = |F|$ be the number of feedback reviews in F. It runs in $O(mn)$ to compare reviews with all bitcoin transactions.

To speed up the search process, we build a 2-D tree for the bitcoin transaction set B where the 2 dimensions are

Algorithm 2: BuildKDTree(B, η, d) [31] // recursive function

Input: Bitcoin transaction set B, max depth η, current depth d
Output: a KD-tree T

if $d < \eta$ **then** // $d < \eta$, a non-leaf node of the KD-tree
  create a KD-tree T with a root node π;
  if d *is odd* **then** // d is odd, split by the timestamp
    $\pi.t \leftarrow$ the median value of all timestamps in B;
    split B into $B_1(t < \pi.t)$ and $B_2(t \geq \pi.t)$ by time;
  else // d is even, split by the money value
    $\pi.u \leftarrow$ the median value of all money values in B;
    split B into $B_1(u < \pi.u)$ and $B_2(u \geq \pi.u)$ by value;
  $T_{\text{left}} \leftarrow \text{BuildKDTree}(B_1, \eta, d+1)$; // build the left KD-tree
  $T_{\text{right}} \leftarrow \text{BuildKDTree}(B_2, \eta, d+1)$; // build the right KD-tree
  add a left child sub-tree T_{left} and a right child sub-tree T_{right} to π;
else // $d = \eta$, a leaf node of the KD-tree
  create a KD-tree T with a single node π containing set B;



(a) coordinate system representation (b) tree representation

Fig. 3. K-D tree and range searching

timestamp and money value. The reason that we build a KD-tree for B, not for F is that the number n of bitcoin transactions is generally much larger than the number m of feedback reviews of a vendor. Algorithm 2 shows a recursive method for building a KD-tree with a fixed height η from set B. In the even depth nodes of the KD-tree, timestamp is used for partitioning the bitcoin transactions. In the odd depth nodes of the KD-tree, money value is used. To build the entire KD-tree, we call "BuildKDTree(B, η, $d = 1$)" in algorithm 2 and pass the entire set B, the maximum depth η of the KD-tree, and the initial depth $d = 1$ to the function. Figure 3(a) shows an example of building a KD-tree and Figure 3(b) shows the resulting KD-tree. In each depth d, Algorithm 2 needs a linear time $O(n)$ to find the median and split the set into left and right subsets. Algorithm 2 runs in $O(\eta n)$.

Algorithm 3 shows the improved bounding box search algorithm using the KD-tree. For each feedback review f, Algorithm 3 will find the leaf nodes in the KD-tree that may contain bitcoin transactions matched with f. Searching a KD tree runs in $O(2^{\eta})$ in the worst case. W represents all leaf nodes of the KD-tree are returned. In general, the number of returned leaf nodes is small. Suppose on average, the number

Algorithm 3: BoundingBoxSearchWithKDTree(B, T, F, η, θ, ϕ_1, ϕ_2)

Input: bitcoin transaction set B, KD-tree T built from B, feedback review set F, max depth η, bounding box parameter θ for timestamp, ϕ_1 and ϕ_2 for money value

Output: a family of sets $\{S_i\}$, where S_i represents the set of feedback reviews matched with r_i in $b_i(t_i, u_i, r_i) \in B$

for each unique r_i in $b_i(t_i, u_i, r_i) \in B$ **do** $S_i \leftarrow \emptyset$; // initialization
$f_j(\tau_j, v_j) \in F : W =$ KDTreeSearch(root node of KD-tree T, $f(\tau, v)$, η, θ, ϕ_1, ϕ_2);
$b_i(t_i, u_i, r_i) \in W$: **if** $\tau_j \leq t_i \leq \tau_j + \theta$ **and** $\phi_1 v_j \leq u_i \leq \phi_2 v_j$ **then**
  $S_i \leftarrow S_i \cup \{f_j\}$;

Algorithm 4: GreedySetCoverAlgorithm($\{S_i\}$, λ)

Input: a family of sets $\{S_i\}$, where S_i represents the set of feedback reviews matched with r_i in $b_i(t_i, u_i, r_i) \in B$, a threshold $\lambda \in (0, 1)$

Output: optimal number k

$r_1 \leftarrow \text{argmax}_{r \in R}\, s(r)$; $R_1 \leftarrow \{r_1\}$; $i \leftarrow 1$; // extract the best matching
$(s(R_i) - s(R_{i-1}))/(s(R_{i-1}) - s(R_{i-2})) < \lambda$
$r_{i+1} \leftarrow \text{argmax}_{r \in R \setminus R_i}(s(R_i \cup \{r\}) - s(R_i))$;
$R_{i+1} \leftarrow R_i \cup \{r_{i+1}\}$, $i++$;
return R_{i-1};

of returned leaf nodes is γ. Since there are 2^η leaf nodes, there are $O(n/2^\eta)$ bitcoin transactions in each leaf node. Therefore, Algorithm 3 runs in $O((\gamma+\eta)mn/2^\eta)$ on average. It is much faster than $O(mn)$. γ is small in our experiments on real data.

B. Greedy Set Cover Algorithm

Algorithm 4 shows a greedy algorithm with a ratio $(1-1/e)$ of optimal for solving Problem 3. In Algorithm 4, we start with an empty address set $R_0 = \emptyset$, and add a bitcoin address r_i in each iteration which maximally increases the review coverage $s(R_i)$.

Theorem 3: Algorithm 4 has an approximation ratio of $(1 - 1/e)$. That is, $s(R_k) \geq (1 - 1/e)s(A^*)$ where A^* represents the k-size address set which matches maximum reviews and k is the size of returned address set.

The proof can be found in [32]. The key observation is that $s(R)$ is non-decreasing submodular set function according to Theorem 1.

Algorithm 4 can decide the number of addresses to return with a threshold λ. Intuitively a vendor's bitcoin addresses should match many more of their feedback reviews compared with noise bitcoin addresses that do not belong to them. This phenomenon is also proved in our experiments on real-life data. We calculate the ratio of increments in review coverage in two consecutive iterations. If the ratio is less than a threshold $\lambda \in (0, 1)$, the greedy algorithm will terminate and output address set.

Algorithm 5: Cost-Effective Addresses Searching(B, F, θ, ϕ_1, ϕ_2, λ)

Input: bitcoin transaction set B, feedback review set F, bounding box parameter θ for timestamp, ϕ_1 and ϕ_2 for money value, a threshold $\lambda \in (0, 1)$

Output: a set R_k of bitcoin addresses belonging to a vendor

$R' \leftarrow \emptyset$; $F' \leftarrow \emptyset$; $\alpha' \leftarrow$ None; // Stage 1: an address with max coverage
$s(\alpha') < |F \setminus F'|$ $F_{\alpha'} \leftarrow$ all feedback reviews matched with bitcoin address α';
$f \leftarrow$ select a feedback review from $F \setminus (F' \cup F_{\alpha'})$ at random;
$R' \leftarrow R' \cup \{$ bitcoin addresses matched with $f\}$;
$F' \leftarrow F' \cup \{f\}$;
$\alpha' \leftarrow \text{argmax}_{r \in R'}\, s(r)$; // extract the best matching address
$R_1 \leftarrow \{\alpha'\}$; $i \leftarrow 1$; // Stage 2: searching for a set of addresses
$(s(R_i) - s(R_{i-1}))/(s(R_{i-1}) - s(R_{i-2})) > \lambda$ $F_{\alpha'} \leftarrow$ all feedback reviews matched with bitcoin address α';
$F \leftarrow F \setminus F_{\alpha'}$; $F' \leftarrow F' \setminus (F' \cap F_{\alpha'})$;
$\alpha' \leftarrow \text{argmax}_{r \in R'}\, s(r)$; // best matching address based on new F
$R_{i+1} \leftarrow R_i \cup \{\alpha'\}$; $i++$;
return R_{i-1};

C. Cost-Effective Addresses Searching

Algorithm 3 will find matched bitcoin transactions for all feedback reviews in F. And the following greedy algorithm 4 will find an optimal address which cover most reviews iteratively. The whole process is time consuming. In this section, we propose a Cost-Effective Addresses Searching(CEAS) algorithm which will find the optimal address with much less matching calculations between reviews and bitcoin transactions. The steps are shown in algorithm 5. In CEAS, we only needs to apply range searching for $(|F| - s(r_{max}) + 1)$ reviews, where r_{max} is the address that covers maximum reviews. Therefore $s(r_{max})$ is the maximum number of reviews matched by a single address. Experiments show that CEAS can prunes about 60% comparisons between reviews and bitcoin transactions.

Let F represents full feedback reviews and F' represents feedback reviews set which we have already applied range searching for. R is the address set containing all addresses and R' represents address set containing addresses matched by any reviews in F'. We have following theorem.

Theorem 4: Let $\alpha = \text{argmax}_{r \in R}\, s(r)$ and $\alpha' = \text{argmax}_{r \in R'}\, s(r)$. If $s(\alpha') \geq |F \setminus F'|$, we have $\alpha' = \alpha$.

α is the address in R that matches the maximum number of reviews and α' is the address in R' that matches the maximum number of reviews. For any bitcoin address $r \in R \setminus R'$, it doesn't match any reviews in F'. Therefore, the maximum number of reviews that r can match is $|F \setminus F'|$. If $s(\alpha') \geqslant |F \setminus F'|$, address α' matches more reviews than any address $r \in$R $\setminus R'$, which makes α' the address that matches largest number of reviews in set R. This completes the proof.

According to Theorem 4, we can find the optimal address

in R' which is also the optimal address we could find in R when the condition $s(\alpha') \geq |F \setminus F'|$ is satisfied. Therefore we don't need to apply range searching in KD-tree for all reviews in R to get this optimal address.

Theorem 5: In all addresses R, to find the optimal address which matches maximum number of reviews in F, The number of feedback reviews $|F'|$ we need to apply range searching for is bounded as $|F| - s(r_{max}) \leq |F'| \leq |F| - s(r_{max}) + 1$.

r_{max} represents the address in R that matches maximum reviews. $s(r_{max})$ represents the number of feedback reviews that are matched with the address r_{max}. To satisfy the condition $s(\alpha') \geqslant |F \setminus F'|$ in theorem 4, we have $|F'| \geq |F| - s(\alpha') \geq |F| - s(\alpha) = |F| - s(r_{max})$. At the same time, α need to match at least one review in F', which requires at most $(|F| - s(r_{max}) + 1)$ times of range searching.

Now we know we need at most $(|F| - s(r_{max}) + 1)$ times of range searching to get the optimal address. In greedy algorithm, we need to repeat this step for k times to obtain the optimal k-size address set. We can prove that finding the optimal address set still need at most $(|F| - s(r_{max}) + 1)$ times range searching.

Theorem 6: To find optimal address set in greedy Algorithm 5, the total number of feedback reviews $|F'|$ we need to apply range searching is still bounded as $|F| - s(r_{max}) \leq |F'| \leq |F| - s(r_{max}) + 1$.

After we apply $|F| - s(r_{max})$ or $|F| - s(r_{max}) + 1$ times of range searching to get the first address. The next address should be the one which matches maximum reviews in the remaining reviews based on greedy algorithm. We can remove reviews covered by the first address and use the same method to find the second address which matches maximum reviews in the remaining reviews. When we select feedback review to apply range searching in step 4 of algorithm 5, we avoid the reviews which are matched with address α'. As a result, in all $|F| - s(r_{max})$ feedback reviews where we apply range searching on, only one feedback review match with the address r_{max}. After we remove reviews covered by the address r_{max}. In the remaining $F - s(r_{max})$ reviews, there should be $|F| - s(r_{max}) - 1$ reviews that have already gone through range searching. Now we can update $F = (F - s(r_{max}))$ and update $F' = (|F| - s(r_{max}) - 1)$. Then the new $|F \setminus F'| = ((F - s(r_{max})) - (|F| - s(r_{max}) - 1) = 1$, which means $s(\alpha') \geq |F \setminus F'|$ in theorem 4 is always satisfied. Therefore we don't need to apply range searching to any more reviews to find remaining addresses of vendor.

Figure 4 explains CEAS Algorithm using the example in Table II. In Figure 4, we use a bipartite graph to represent the matches between feedback reviews and receiver addresses. Each node on the left represents a feedback review and each node on the right represents a receiver address. An edge represents that a feedback review is covered by this address. A dotted edge represents a non-computed match and a solid edge represents a computed match. We re-order the nodes on both sides to reduce the visual clutter.

In Stage 1, Algorithm 5 randomly selects f_1 in line 4 and finds its matched bitcoin addresses $\{r_2, r_3, r_4, r_5, r_6\}$ in line 5. Now $F' = \{f_1\}$ and $R' = \{r_2, r_3, r_4, r_5, r_6\}$. We use black nodes to represent them in Figure 4(a). Algorithm 5 extracts the bitcoin address r_3 from R' since it has the largest number of matched feedback reviews. In Figure 4(b), we use "$*$" to represent the bitcoin address in R' with the maximum coverage. Now $F_{\alpha'} = \{f_1, f_4, f_5, f_6\}$. Algorithm 5 then randomly selects f_2 from the review set $F \setminus (F' \cup F_{\alpha'}) = \{f_2, f_3, f_7\}$ in line 4 and finds its matched bitcoin addresses $\{r_1, r_7\}$ in line 5. Now $R' = \{r_1, r_2, r_3, r_4, r_5, r_6, r_7\}$ and $F' = \{f_1, f_2\}$. Figure 4(c) shows the status. Algorithm 5 extracts the bitcoin address r_1 from R' since it has the largest number of matched feedback reviews, which is shown in Figure 4(d). Since $s(r_{max}) = s(\{r_1\}) = 5 \geq |F \setminus F'| = 5$, Stage 1 is done.

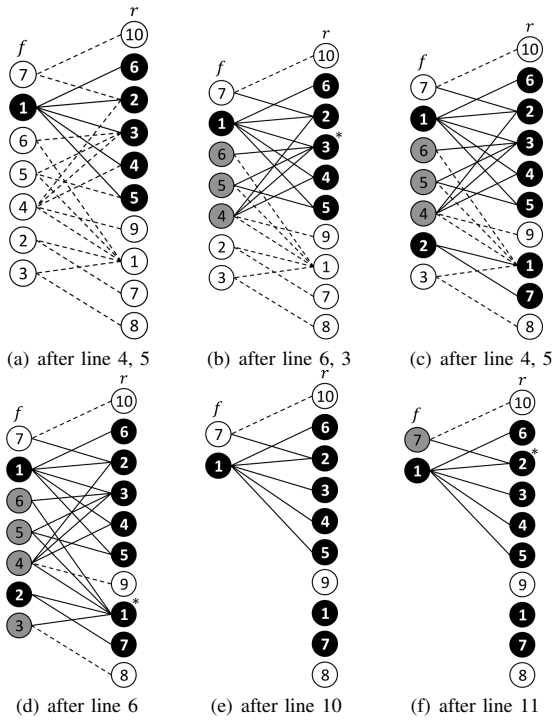


Fig. 4. An example for the heuristic search in Algorithm 5 (F: nodes on the left; F': black nodes on the left; R': black nodes on the right; the "line" refers to the lines in Algorithm 5)

In Stage 2, Algorithm 5 first adds the best bitcoin address r_1 into the resulting set thus $r_{max} = r_1$. Since r_1 is matched with $F_{r_1} = \{f_2, f_3, f_4, f_5, f_6\}$, Algorithm 5 deletes the reviews covered by r_1 from the left part and the associated edges. Figure 4(e) shows the remaining graph. Now $F = \{f_1, f_7\}$ and $F' = \{f_1\}$. Algorithm 5 then selects the best matching address from R' based on the new $F = \{f_1, f_7\}$. Since r_2 has the largest number of matches thus is optimal. Algorithm 5 will terminate in the next iteration since all feedback reviews have been covered and the drop is larger than λ.

Here we only need to search matched bitcoin transactions for f_1 and f_2. This is one possible solution for this example. In algorithm 5, we randomly select review to apply range searching. No matter how we select reviews, it always needs to apply range searching for 2 or 3 reviews in this example, which

is the range of $|F'|$. In this example,$2 \leq |F'| \leq 3$ because $s(r_{max}) = 5$ and $|F| = 7$. In contrast, Algorithm 3 need apply range searching for all 7 feedback reviews. According to Theorem 4, Algorithm 5 always finds the same best bitcoin address that Algorithm 4 finds.

Time Complexity: Line 5 in Algorithm 5 performs the bounding box search and is the most time consuming step and dominates the time complexity. This is because the number of bitcoin transactions n is much larger than other parameters like m. Based on Theorem 5, the times of running line 5 is upper bounded by $|F'| \leq m - s(r_{max}) + 1$. Thus Algorithm 5 runs in $O((\gamma + \eta)(m - s(r_{max}) + 1)n/2^{\eta})$. Please refer to last paragraph in Section IV-A for more details.

V. Experimental Results

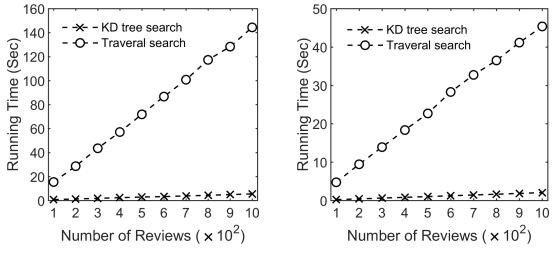
In this section, we conduct extensive experiments to evaluate the efficiency and effectiveness of our method by using both real and synthetic datasets. All algorithms are implemented in Python. All the experiments are conducted on a Linux Server with Intel Xeon 3.2GHz CPU and 32 GB main memory.

Datasets. For the real dataset, we crawled feedback from Wall Street Market. Wall Street Market sells a variety of content, including drugs, stolen data, and counterfeit consumer goods, all using cryptocurrency. In the Wall Street market, each vendor has a list of reviews. Each feedback contains the time when the buyer leaves this feedback as well as the amount of Bitcoin used in this transaction. Here we crawled the feedback of different vendors and sum the transactions to a file, one transaction was represented by a 2-dimensional data (τ, v), where τ is the timestamp when this review was posted and v is money cost in this purchase. Here, we collect transactions of different vendors in Dec 2018. There are in total 17,155,754 bitcoin transactions during this time. The synthetic dataset is produced by the Bitcoin transaction data.

Next, we first evaluate the efficiency of K-D tree and greedy algorithm by real dataset and then the accuracy by synthetic dataset.

A. Efficiency Evaluation in Range Searching

Each review we have can generate a 2-dimensional range based on the Bitcoin value and timestamp. With this range, we search the Bitcoin public ledger to find a lot of candidate transactions matched to this review. Figure 5 demonstrates that the K-D tree we build can effectively save time during range searching. We build a K-D tree with the real dataset, separating transactions into 16,384 buckets with a 14-depth binary tree. Each bucket contains more than one thousand transactions. We sample reviews from Wall Street Market. Bitcoin transactions are divided through Bitcoin value and timestamp alternately. Figure 5(a) shows the comparison of time-consuming in K-D tree and traversal in full ledger. It only takes less than 5.5 seconds to find the matched transactions for 1000 reviews in the K-D tree, nearly 5.5 milliseconds for one review. We also build another K-D Tree with a smaller data size, including only mixed transactions in Bitcoin public ledger. Users in cryptomarkets prefer mixed transactions to protect their privacy in Bitcoin transactions. We conduct the same experiment in this filter ledger which contains 1,395,694 bitcoin transactions. Figure 5(b) shows the result with K-D tree structure, it only takes around 2 milliseconds to find the matched transactions for a review.



(a) range searching in full ledger (b) range searching in filtered ledger

Fig. 5. Compare running time of K-D Tree searching and traversal searching in full ledger and filter ledger with only mix transactions

B. Effectiveness Evaluation of Greedy Algorithm

Our greedy algorithm guarantees that we can achieve at least $(1 - 1/e)$ of maximum coverage theoretically. Here we speed up our greedy algorithm by removing low degree addresses found in range searching and evaluate the performance of the greedy algorithm on the real dataset.

We select feedback of 100 vendors with 3721 reviews. Matched bitcoin transactions of reviews from a vendor can be found through the K-D tree, which helps us get matched addresses of each review. By changing these data format to vertical format like Table III, we get the reviews covered by each matched address for a vendor. Now we are looking for a receiver address set whose transactions can match the maximum reviews. Based on the range we set, a review can normally be matched to thousands of transactions in the Bitcoin ledger. Only one of these matched addresses can be the vendor's address, which means the remaining addresses are noises in our algorithm. In the experiments, 94.23% of the addresses we found only match one review. 5.26% of addresses match 2 reviews and only 0.51% addresses match more than 2 reviews on average. Heuristically we are looking for addresses that can match the maximum reviews. Therefore, removing addresses with a low degree will not affect the accuracy of our algorithm. We conduct the simple greedy algorithm without setting a threshold λ. We set k from 1 to 10 as the number of output addresses. The greedy algorithm needs to output an address set that covers as many reviews as possible. Result demonstrates that we can save 93% time if we ignore address matching only 1 review during the greedy algorithm and 99% time if we remove addresses with a degree less than 3.

To evaluate the performance in the maximum coverage of the greedy algorithm. We compare the greedy algorithm with a heuristic high degree method and the random selection method. The high degree method will select addresses with the maximum review coverage. We average the percentages of reviews covered, From Figure 6, we can notice that the performance of high degree method is similar to the greedy

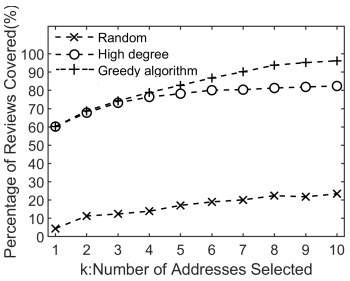


Fig. 6. Greedy algorithm outperform high degree and random method

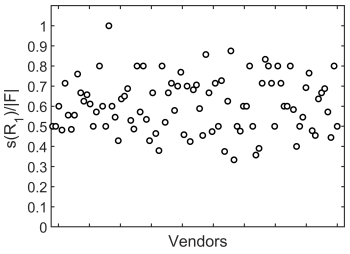


Fig. 7. Maximum reviews covered by one address in ratio for different vendors

algorithm when k is small. As k increases, the gap between these two methods increases.

The number of times in range searching is at most $|F| - s(R_1) + 1$ in our proposed algorithm. We can reduce at least $s(R_1) - 1$ times. $s(R_1)$ is the number of reviews covered by the first address we get in the greedy algorithm, which also is the address r_{max} that covers maximum reviews in F. The more reviews covered by this address, the fewer times of range searching are required. Figure 7 shows the ratio $s(R_1)/F$ of 100 vendors we find in Wall Street Market. Each node in Figure 7 represents a vendor. We can see that the address which covers the maximum reviews can cover between 35% to 90% of all reviews from a vendor and 60.217% on average, which means we can reduce times of range searching by 60.217%. We find 66% of these vendors whose reviews can be matched to an address that covers more than half of the reviews. The result shows the effectiveness of our algorithm in the real dataset and unveils that the vendors in Wall Street Market do not change their receiving addresses frequently.

C. Accuracy Evaluation on Synthetic Data

In this section, we conduct experiments to evaluate the accuracy of our algorithm on synthetic reviews. We select 2000

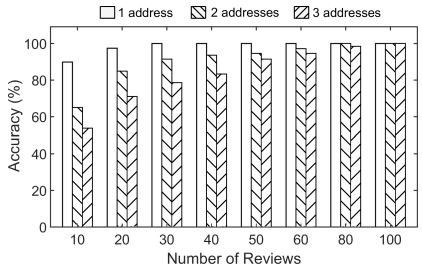


Fig. 8. Accuracy comparison between vendors with different number of addresses

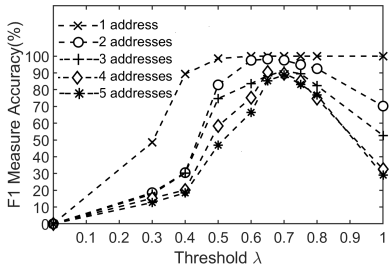


Fig. 9. F1 Measure accuracy of synthetic data generated from different number of addresses

Bitcoin addresses and collect their history transactions. Each transaction contains timestamp and Bitcoin value received, which can be treated as a review after a slight change. We apply normal distribution on the amount of change in both Bitcoin value and timestamp. The number of hours we advance on timestamp follows the normal distribution with 0.5 mean and 0.6 std. In probability, the newly generated review's bounding box in timestamp will cover the transaction at 59% feasibility. The same strategy is applied to the Bitcoin value. After combining the restrict of timestamp and Bitcoin value, the newly generated review can match to the original transaction with a 35.4% possibility. Every address we randomly select from the Bitcoin ledger can generate a synthetic review list. Considering vendors may use multiple Bitcoin addresses, we also combine some synthetic reviews generated by different addresses.

Figure 8 shows the accuracy of the greedy algorithm with reviews generated by 1 address, 2 addresses, and 3 addresses. Accuracy is the number of correct addresses over the number of addresses that generate these synthetic reviews. From Figure 8, we can see that longer review lists contribute to better accuracy, while more receiver addresses can reduce accuracy, which matches the real situation. It is hard to find the vendor's address set if the vendor updates their receiver address in the darknet market very frequently. For vendors who do not change their receiver address frequently, our algorithm can achieve great performance even with very few reviews.

We set a threshold λ for the ratio of new reviews covered in the current step to new reviews covered in the last step. We use synthetic data generated by different numbers of addresses to evaluate the effect of different λ. For reviews generated from one address, Figure 9 shows larger λ has better performance because we do not want to select another address besides the one with the highest degree. Reviews derived from 2 or more addresses share a similar pattern. Large λ can decrease the accuracy because a high threshold will stop the greedy algorithm too early and output fewer addresses than the vendor have. The more addresses a feedback related, the fast the drop of accuracy after λ pass 0.7. Through the experiments, we can see λ around 0.7 is the best option for all these data.

VI. Conclusion

In this paper, we study the problem of identifying the bitcoin addresses of a vendor by matching their feedback

reviews with bitcoin transactions. We firstly construct a K-D tree to efficiently match Bitcoin transactions to feedback reviews. After we obtain the matching relationship between bitcoin transactions and feedback reviews, we get the address set by applying a greedy algorithm that can achieve near-optimal with theoretical guarantee. We further develop a Cost-Effective Address Searching(CEAS) algorithm that can speed up the process by pruning the search space effectively. Comprehensive experiments on both real and synthetic datasets demonstrate the effectiveness and efficiency of our methods.

VII. Acknowledgement

This work is partially supported by the DHS/CINA project entitled "Open Source Intelligence in Online Stolen Data Markets: Assessment of Network Disruption Strategies", and the NSF awards 2030636 and 2039949.

References

[1] J. Van Buskirk, S. Naicker, A. Roxburgh, R. Bruno, and L. Burns, "Who sells what? country specific differences in substance availability on the agora cryptomarket," *International Journal of Drug Policy*, vol. 35, pp. 16–23, 2016.

[2] S. Kethineni, Y. Cao, and C. Dodge, "Use of bitcoin in darknet markets: Examining facilitative factors on bitcoin-related crimes," *American Journal of Criminal Justice*, vol. 43, no. 2, pp. 141–157, 2018.

[3] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Manubot, Tech. Rep., 2019.

[4] F. Reid and M. Harrigan, "An analysis of anonymity in the bitcoin system," in *Security and privacy in social networks*. Springer, 2013, pp. 197–223.

[5] R. Böhme, N. Christin, B. Edelman, and T. Moore, "Bitcoin: Economics, technology, and governance," *Journal of economic Perspectives*, vol. 29, no. 2, pp. 213–38, 2015.

[6] S. Lee, C. Yoon, H. Kang, Y. Kim, Y. Kim, D. Han, S. Son, and S. Shin, "Cybercriminal minds: an investigative study of cryptocurrency abuses in the dark web," in *Network and Distributed System Security Symposium*. Internet Society, 2019, pp. 1–15.

[7] S. Underwood, "Blockchain beyond bitcoin," 2016.

[8] G. Hileman and M. Rauchs, "Global blockchain benchmarking study," *Rochester, NY: Social Science Research Network*, 2017.

[9] X. Chen, M. Al Hasan, X. Wu, P. Skums, M. J. Feizollahi, M. Ouellet, E. L. Sevigny, D. Maimon, and Y. Wu, "Characteristics of bitcoin transactions on cryptomarkets," in *International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage*. Springer, 2019, pp. 261–276.

[10] Y. Fanusie and T. Robinson, "Bitcoin laundering: an analysis of illicit flows into digital currency services," *Center on Sanctions and Illicit Finance memorandum, January*, 2018.

[11] M. Tran, L. Luu, M. S. Kang, I. Bentov, and P. Saxena, "Obscuro: A bitcoin mixer using trusted execution environments," in *Proceedings of the 34th Annual Computer Security Applications Conference*, 2018, pp. 692–701.

[12] M. Spagnuolo, F. Maggi, and S. Zanero, "Bitiodine: Extracting intelligence from the bitcoin network," in *International Conference on Financial Cryptography and Data Security*. Springer, 2014, pp. 457–468.

[13] M. Fleder, M. S. Kester, and S. Pillai, "Bitcoin transaction graph analysis," *arXiv preprint arXiv:1502.01657*, 2015.

[14] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, and S. Capkun, "Evaluating user privacy in bitcoin," in *International Conference on Financial Cryptography and Data Security*. Springer, 2013, pp. 34–51.

[15] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage, "A fistful of bitcoins: characterizing payments among men with no names," in *Proceedings of the 2013 conference on Internet measurement conference*, 2013, pp. 127–140.

[16] D. Genkin, D. Papadopoulos, and C. Papamanthou, "Privacy in decentralized cryptocurrencies," *Communications of the ACM*, vol. 61, no. 6, pp. 78–88, 2018.

[17] M. Masoni, M. R. Guelfi, and G. F. Gensini, "Darknet and bitcoin, the obscure and anonymous side of the internet in healthcare," *Technology and Health Care*, vol. 24, no. 6, pp. 969–972, 2016.

[18] S. Kethineni, Y. Cao, and C. Dodge, "Use of bitcoin in darknet markets: Examining facilitative factors on bitcoin-related crimes," *American Journal of Criminal Justice*, vol. 43, no. 2, pp. 141–157, 2018.

[19] R. Upadhyaya and A. Jain, "Cyber ethics and cyber crime: A deep dwelved study into legality, ransomware, underground web and bitcoin wallet," in *2016 International Conference on Computing, Communication and Automation (ICCCA)*. IEEE, 2016, pp. 143–148.

[20] R. S. Portnoff, D. Y. Huang, P. Doerfler, S. Afroz, and D. McCoy, "Backpage and bitcoin: Uncovering human traffickers," in *KDD*, 2017, pp. 1595–1604.

[21] J. E. Hopcroft and R. M. Karp, "An nˆ5/2 algorithm for maximum matchings in bipartite graphs," *SIAM Journal on computing*, vol. 2, no. 4, pp. 225–231, 1973.

[22] J. Edmonds, "Maximum matching and a polyhedron with 0, 1-vertices," *Journal of research of the National Bureau of Standards B*, vol. 69, no. 125-130, pp. 55–56, 1965.

[23] R. Jonker and T. Volgenant, "Improving the hungarian assignment algorithm," *Operations Research Letters*, vol. 5, no. 4, pp. 171–175, 1986.

[24] A. Krause and D. Golovin, "Submodular function maximization." 2014.

[25] J. Edmonds, "Submodular functions, matroids, and certain polyhedra," in *Combinatorial Optimization—Eureka, You Shrink!* Springer, 2003, pp. 11–26.

[26] R. K. Iyer and J. A. Bilmes, "Submodular optimization with submodular cover and submodular knapsack constraints," in *Advances in Neural Information Processing Systems*, 2013, pp. 2436–2444.

[27] U. Feige, V. S. Mirrokni, and J. Vondrák, "Maximizing non-monotone submodular functions," *SIAM Journal on Computing*, vol. 40, no. 4, pp. 1133–1153, 2011.

[28] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *KDD*, 2003, pp. 137–146.

[29] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, "Cost-effective outbreak detection in networks," in *KDD*, 2007, pp. 420–429.

[30] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of computer computations*. Springer, 1972, pp. 85–103.

[31] J. L. Bentley and J. H. Friedman, "Data structures for range searching," *ACM Computing Surveys (CSUR)*, vol. 11, no. 4, pp. 397–409, 1979.

[32] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—i," *Mathematical programming*, vol. 14, no. 1, pp. 265–294, 1978.