A Bitcoin Transaction Network Analytic Method for Future Blockchain Forensic Investigation

Yan Wu, Fang Tao, Lu Liu, *Member, IEEE*, Jiayan Gu, John Panneerselvam, Rongbo Zhu, Mohammad Nasir Shahzad

***Abstract*— Popular Blockchain-based cryptocurrencies, like Bitcoin, are increasingly being used maliciously to launder money on the dark Web. In order to trace and analyze suspected Bitcoin transactions and addresses, address clustering methods and Bitcoin flow analysis methods are gaining attention recently. However, existing methods only focus on Bitcoin addresses and flow, and neglect other important information, such as transaction structure and behavior features. In order to exploit all useful features of transactions, this paper proposes a Bitcoin transaction network analytic method for facilitating Blockchain forensic investigation based on an extended safe Petri Net. The structural features and dynamic semantics of Petri net are used in our proposed model to define the static and dynamic features of Bitcoin transactions. Nineteen features have been identified to define Bitcoin transaction patterns for analyzing and finding suspected addresses. Bitcoin gene has been embedded into the Petri net transitions to trace and analyze Bitcoin flow accurately. Finally, marginal distribution analysis of Bitcoin transaction features and data visualization techniques are used to eliminate some false positive samples further and to improve the accuracy of identifying suspected addresses. The proposed Bitcoin transaction network analytic method provides a reliable forensic investigation model along with a prototype platform which is beneficial for financial security. The efficiency of our proposed method is empirically verified based on a real-life case study analysis.**

***Index Terms*—Bitcoin, Blockchain, Petri Net, Forensic Investigation.**

I. Introduction

BITCOIN has become increasingly popular as an alternative form of currency over the last few years [1], and its growth has been inevitable since its introduction by Satoshi Nakamoto. The market capitalization of Bitcoin has been witnessed to have reached more than $200 billion at the end of 2017. Bitcoins are not usually associated with their user identities such as user names, residential addresses, or other personal identification information. Due to this pseudonym nature, Bitcoin is falsely regarded as a form of anonymous currency in the Internet, and is falsely believed to be facilitating untraceable transactions during illegal trades [2].

Since its pseudonym nature, Bitcoin has soon been used by illegal activities, such as illegal drugs and weapon trade etc. Silk Road [3], which was an online black market and the first modern dark net market initiated in 2011, has witnessed an increased illegal usage of Bitcoins. It used Bitcoin as a major source of transactions, an estimated $15 million dollars transactions have been recorded during the period of 3 February 2012 to 24 July 2012. AlphaBay and Hansa [4] are two of the biggest “dark web” contraband marketplaces, used for illegal sale of guns, drugs, pharmaceuticals, forged documents, stolen card details and other forbidden merchandise, which has also witnessed an increased use of Bitcoin for illegal trades. Even enforcement agencies found it really challenging to ban and lock such dark net markets and so the illegal usage of Bitcoins. It is worthy of note that almost 95% of the laundered coins are linked to the nine dark-web marketplaces. An operator of BTC-e [5], which is an exchange used to trade Bitcoins since 2011, laundered more than $4 billion worth of illegal funds by people involved in crimes ranging from computer hacking to drug trafficking. Terrorists have also been associated with Bitcoin as early as 2012, using Bitcoin for illegal fund transfers and donations. Moreover, Bitcoin is not recognized as lawful electronic currencies at present, thus defined regulations for using Bitcoin are hardly existing till now. Ajello [6] highlighted the importance of anti-money laundering of Bitcoin from a legal aspect. Similarly, Perri Reynolds and Angela S.M. Irwin [7] discussed the necessity of Bitcoin tracing and analysis from a legal perceptive.

This necessitates an in-depth analysis of Bitcoin transactions for the purpose of detecting and acting against illegal transactions. Bitcoin transactions are stored on Bitcoin blockchain publicly. It means if a criminal Bitcoin address is known, Bitcoins passed the address can be tracked. Bitcoin exchanges are usual places of changing Bitcoins to fiat currencies. Generally, Bitcoin exchanges are required by "know your customer" law to collect personal information. If some coins are found being transferred from a suspicious address into an exchange's address, the identity of the suspect can be found. Therefore, analyzing the suspected addresses is very important for identifying suspects [8].

Tracking Bitcoins associated with a known address is not

- *Yan Wu and Fang Tao are with the School of Computer Science and Telecommunication Engineering, Jiangsu University, Jiangsu, China and and also with Jiangsu Key Laboratory of Security Technology for Industrial Cyberspace, Jiangsu University, China*. (e-mails: wuyan04418@ujs.edu.cn; taofang7906@hotmail.com).
- *Lu Liu (corresponding author) and Mohammad Nasir Shahzad are with the School of Informatics, University of Leicester, UK*. (e-mails: l.liu@leicester.ac.uk; mns14@leicester.ac.uk)
- *Jiayan Gu and John Panneerselvam are with the School of Electronics, Computing, and Mathematics, University of Derby, UK*. (e-mails: j.gu2@unimail.derby.ac.uk; j.panneerselvam@derby.ac.uk)
- *Rongbo Zhu is with the College of Computer Science, South-Central University for Nationalities, China*. (e-mail: rbzhu@mail.scuec.edu.cn)

usually an issue. However, tracking Bitcoins has been complicated, since criminal's addresses are often cloudy and uncertain. To this end, this paper is aimed at distinguishing suspected addresses based on common transaction patterns and features. In general, some transactions might exhibit similarities and common patterns. For example, Bitcoin transactions being used to gather Bitcoins usually associate multiple input addresses and an output address. Analyzing the links between such input and output addresses could present useful insights when tracking unknown and suspected transactions. But such an analysis includes other complexities such as defining the Bitcoin transaction features, effectively identifying the features those can provide useful and meaningful information whilst identifying suspects, effectively tracking Bitcoins passed by multiple suspected addresses, and importantly resolving such analytics requirements with a reasonable time-scale to allow necessary actions, despite the nature of massive volumes of Blockchain data.

With this in mind, this paper proposes an extended safe Petri net [9] based model to simulate the Bitcoin transactions, which is called Bitcoin Transaction Net (BTN). Its structural features and dynamic semantics are used to describe both the static and dynamic features of Bitcoin transactions, respectively. Nineteen static and dynamic Bitcoin transaction features have been identified to define Bitcoin transaction patterns for analyzing and finding suspected addresses by our pattern matching method. Another key contribution of our method is the development of the Bitcoin gene that is embedded into Petri net transitions. Three gene operations, called merging, splitting and dyeing, are defined to evolve genes when a transaction occurs. Bitcoin gene indicates whether an address has a relationship with some specific addresses, along with describing the strength of such relationships. Bitcoin gene can be efficiently used to trace and analyze the flow of Bitcoins easily and accurately. Furthermore, this paper proposes a set of match rules to find transactions and obtain suspected addresses, based on the combinations of match rules. Finally, a marginal distribution analysis of transaction pattern features is incorporated into the proposed model to remove part of false positive samples and enhance the accuracy of the identified suspected addresses.

The remainder of the paper is arranged as follows: Section II presents an analysis of related works based on categorizing Bitcoin transaction analysis methods. Section III describes our proposed framework to formalize transaction patterns and to analyze Blockchain data. Section IV presents the formal modelling of Bitcoin Transaction Net and Section V presents the static and dynamic transaction features. Section VI presents our methodology of analyzing Bitcoin Transaction Pattern and Section VII details the pattern analysis, along with presenting a case study in Section VIII. Section IX concludes this paper along with outlining our future research directions.

II. Related Work

Early studies of the Bitcoin transaction analysis mainly focused on Bitcoin address cluster analysis that aims to cluster addresses owned by a user or an entity. For the forensic analysis, if a Bitcoin address *a* is suspected, and its owner is unknown. However, if the owner of another Bitcoin address *b* is known and *b* is located within the same cluster of *a*, then owner of *a* can be deduced. Reid and Harrigan [8] partitioned addresses into a cluster when these addresses are used as inputs of a transaction. For example, if addresses *a* and *b* are used as inputs of transaction t_1, *a* and *b* are clustered into a single cluster. If addresses *b* and *c* are used as inputs of transaction t_2, then *c* is clustered into the same cluster of *a* and *b*. This input address clustering method is used by many studies [10-14]. Ron and Shamir [10] used this clustering technique to create a contracted transaction graph to analyze the Bitcoin flow. Fleder *et.al.*[11] used the Bitcoin address clustering method to construct a user graph and used PageRank to find important users. Another address clustering method called change address clustering method or shadow address clustering method has also been widely used to collect back the "change" resulted from any transaction issued by the user (input addresses). Androulaki *et.al.*[12] used the input address clustering method and shadow address clustering method to cluster addresses. Meiklejohn *et.al.*[13] also used the input address clustering method and the change address clustering method to cluster addresses and evaluated the accuracy of the change address clustering method. Spagnuolo *et.al.*[14] presented a modular framework, called BitIodine, which parses the blockchain, clusters addresses that are likely to belong to a same user or group of users, and visualizes complex information extracted from the Bitcoin network. BitIodine also uses the two clustering methods to cluster addresses. Meiklejohn *et.al.*[13] pointed out that the change address clustering method has been less robust in spite of the changing patterns within the network, because it is not easy to identify the change addresses. Harrigan and Fretter [15] exhibited the efficiency of the input address clustering method. Therefore, in our framework, only the input address clustering method is used. The function of address clustering in forensic analysis is that once an address is suspected in a cluster, other addresses are also suspected because they are very likely to belong to the same user or group. However, address clustering method cannot identify and analyze transaction pattern.

Another aspect of Bitcoin transaction analysis is the Bitcoin flow analysis. Many existing flow analysis methods [16-19] have used the clusters of addresses as vertexes, and transaction relationships between vertexes as direction edges. Bitcoins usually flow along edges from vertexes to vertexes. Zhao and Guan [16] proposed a classic Bitcoin flow analysis method, where they firstly clustered Bitcoin addresses and then connected the clusters by transaction relationships. Finally, the graph has been analyzed by the visualization methods and statistical methods. Maesa *et al* [17] further used this graph to analyze and verify the assumption whether "the rich get richer". Maesa *et al* [18] used the graph to analyze classical graph properties like densification, distance analysis, degree distribution, clustering coefficient and several centrality measures, but they have not addressed the Bitcoin trace issue. Ober *et al* [19] used the graph to analyze the features of the graph structure that could affect anonymity. However, all such works have not given enough emphasis on analyzing Bitcoin

transaction patterns.

Petri net has been used to analyze Bitcoin transactions [20, 21]. In the Petri net model, Bitcoin addresses are modeled as the places in Petri net and Bitcoin transactions are modelled as transitions in Petri net. Pinna *et al* [20] used input address clustering method in the Petri net model to cluster addresses and found common behavior pattern, such as one-time usage of a given address etc. Pinna [21] used Petri net to analyze disposable addresses which are the addressed used only once. The works of [21] postulated that transactions form chains and the lengths of these chains are characterized by a power-law distribution. These two models used the Bitcoin addresses as Petri net places/inputs of Bitcoin transactions. However, in the Bitcoin transaction net, inputs of Bitcoin transactions are not usually the addresses, but they are coins. Therefore, such models cannot efficiently analyze and quantify transaction features. This paper proposed an extended form of Petri nets for the Bitcoin transaction analysis, which organically integrates important bitcoin features, such as transaction, input and output, address, bitcoin quantity, transaction time and so on.

In contrast to the existing methods which aim to find behavior features behind Bitcoin transactions, our method is to define transaction patterns by transaction features and to find suspected addresses based on the patterns. Monaco [22] presented and verified an assumption that identifying and verifying Bitcoin users based on the observation of Bitcoin transaction features over time holds true. Based on analyzing the behavioral features using 366 user samples, this work concluded that the behavioral patterns observed over time can be used to deprive a user, but this is not further developed into a model. Similar to Monaco [22], Harlev et al. [23] collected 434 training samples and used the supervised machine learning method to classify unidentified entities into known classes. However, in the real world, criminals intend to hide their Bitcoin addresses. It is difficult to find their addresses in order to analyze their transaction features, which limits their practical applicability due to the lack of known samples. Different from their methods, our proposed method does not need such known samples. Our proposed method can use any known information such as information from information agencies to define a transaction pattern and to find addresses matching the pattern.

Data Visualization methods have also been used for Bitcoin transaction analysis. Moser *et al* [24] tested the anti-tracing effectiveness of coin mixing services. Battista *et al* [25] developed a data visualization tool called BitConeView to present the effectiveness of coin mixing services. Christin [26] performed a comprehensive measurement analysis of Silk Road related data collected through web crawling. These visualization methods have been used to illustrate the data characters. Kondor *et al* [27] and Maesa *et al* [18] analyzed the structure of the Bitcoin transaction network by measuring the network characteristics and presented the results through various visualization methods. McGinn *et al.* [28] presented a systemic top-down visualization of Bitcoin systems, which can find the transaction patterns in a block through visual perception. But this method cannot find small transaction patterns behind massive transactions. Bistarelli *et al* [29] developed a tool called BlockChainVis that employs techniques from visual analytics to filter out undesired information in order to visually analyze the transactions. BlockChainVis allows users to define simple rules to filter out undesirable information. [24-26]focused on the visualization of some features of the Bitcoin transactions for specific purposes, while [18, 27, 28] tended to visualize the overall Bitcoin Blockchain. Therefore, some details are easy to be neglected. The works of [29] added customizable filters to find some specific transactions or addresses. Similar to [22], this method considered transaction features separately and ignored defining a transaction pattern. Visualization methods are usually dependent on visual perception to find results. However, due to the limitation of human brain capacity, some results tend to be neglected. In our method, matching of transaction pattern is processed by the pattern matching algorithm, not by visual perception. Therefore, it is more efficient, and hardly neglect details. Our proposed method uses visualization techniques to visualize marginal distributions of various transaction features to filter part of false positive samples, but not to analyze transaction features directly. Therefore, our visualization method does not inherit the drawbacks of the aforementioned visualization methods.

[30] is one of our preliminary work presented in a conference paper that studied mappings between bitcoin transactions and Petri Net and mappings between transaction features and Petri Net properties. It provides feasibility for the method proposed in this paper.

III. The Framework of the Proposed Method

Our proposed method uses clues to formalize a transaction pattern and obtains addresses related to the pattern by analyzing Blockchain data. Fig. 1 presents the framework of our proposed method.

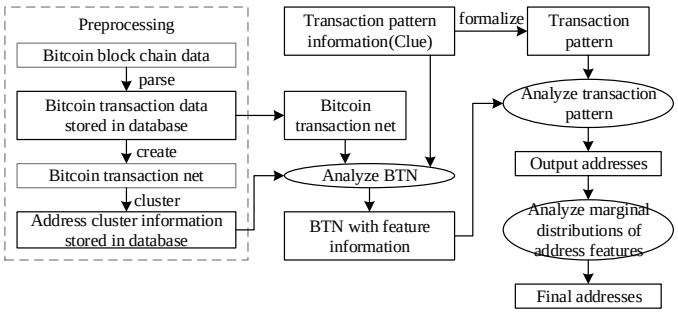


Fig. 1. The framework of the proposed method.

Our proposed framework includes a pre-processing procedure. It contains three steps:

1) The first step is to parse transaction data from Bitcoin Blockchain data and save parsed bitcoin transaction information to a database. An open source tool called BitcoinDatabaseGenera-tor [31] is used to save the data onto the database;

2) The second step is to read the transaction data from the database and to create a Bitcoin Transaction Net (BTN); The creation procedures is introduced in section IV;

3) The third step is to cluster Bitcoin addresses and store the cluster information in the database. The input address clustering

method used by [8, 10-14] is adopted in our framework. The pre-processing procedure can be processed incrementally when new blocks are generated. The pre-processing results can be used by different case's analysis.

When analyzing a case, the BTN needs to be created by reading the transaction data from the database. Next step is to analyze the BTN to obtain various features. In this step, case information (clue) and address cluster information are used to set up the initial state of bitcoin gene for Bitcoin tracing. Petri Net analysis technics are used to analyze the BTN. After the analysis, a BTN with various feature values is obtained.

After that, transaction pattern information (clue) is formalized. The formalized transaction pattern is matched and analyzed with the BTN with feature values. Some addresses related to the pattern are delivered as output. Finally, the marginal distribution analysis method is used to analyze the features of output addresses to eliminate some false positive samples.

IV. Formal Modeling of Bitcoin Transactions

A. Bitcoin transactions

Bitcoin transactions are stored in a growing chain of blocks. There are two types of Bitcoin transactions such as coinbase transactions and regular transactions. A coinbase transaction generates Bitcoins. It has no input, but has at least one output. A regular transaction transfers coins between addresses. It has at least an input and at least an output. Outputs of a given Bitcoin transactions are usually considered as the inputs of the following transactions. A transaction output contains some Bitcoins that are locked by an address. Users can spend the Bitcoins if they have the private key of the address.

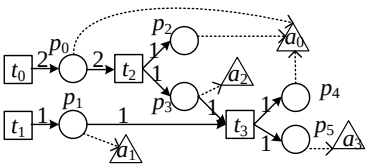


Fig. 2. A simple presentation of connected Bitcoin transactions that contains 4 transactions, 6 outputs and 4 addresses.

Fig. 2 illustrates an example of simplified and connected Bitcoin transaction. It contains 4 transactions (t_0, t_1, t_2 and t_3, among them, t_0 and t_1 are coinbase transactions), 5 outputs (p_0, p_1, p_2, p_3, p_4 and p_5) and 4 addresses (a_0, a_1, a_2 and a_3). p_0, p_1, p_2, p_3, p_4 and p_5 contain 2, 1, 1, 1, 1 and 1 Bitcoins, respectively. They are locked by a_0, a_1, a_0, a_2, a_0 and a_3 respectively. Note that an output can only be locked by an address; an address can lock many outputs.

B. Bitcoin transaction net

An extended safe Petri net based formal model is proposed to describe connected Bitcoins transactions so that Bitcoin transactions can be analyzed through Petri net static and dynamic properties.

Definition 1. Formally, a BTN for a given list of Bitcoin transactions is an 8-tuple $N=(P, T, F, A, \beta, \gamma, \tau, M_0)$ where P, T and A denote finite sets of places (Bitcoin transaction outputs/inputs), transitions (Bitcoin transactions), and Bitcoin addresses, respectively. $F\subseteq(P\times T)\cup(T\times P)$ is a set of arcs between places and transitions. $\beta: P\to R$ is a value function on places, where R denotes real numbers, and $\beta(p)$ is a number of Bitcoin quantity locked in the place P (transaction output). $\gamma: P\to A$ is an address mapping function on places and $\gamma(p)$ denotes an address associated with p. $\tau: T\to T$ is a timestamp function on T, where T denotes timestamps, and $\tau(t)$ denotes timestamp of Bitcoin transaction t. M_0 is an initial marking, where $M_0(p)$ is the number (0 or 1) of tokens in place p.

Coinbase transactions do not have inputs, while a transition without input place in Petri nets is always fireable. In order to limit the fireable numbers of these transitions to 1, an input place is added to each coinbase transactions so that Bitcoin transactions can be analyzed with standard Petri net semantics. These added places are mapped into unique virtual addresses, respectively. The Bitcoin quantity of an added place of a coinbase transaction is the sum of Bitcoin quantities of its output places. In an initial state, if p has no input transition (i.e., p is an input place of a coinbase transaction), then $M_0(p)=1$, otherwise $M_0(p)=0$.

Consequently, the transactions shown in Fig. 2 can be constructed as a BTN shown in Fig. 3. p_6 and p_7 are the newly added places with a token, respectively. Their Bitcoin quantities are 2 and 1, respectively. They are mapped to addresses a_4 and a_5, respectively.

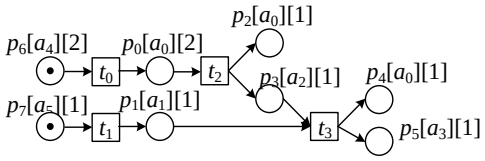


Fig. 3. The BTN of the Bitcoin transaction

For convenience of analysis, Table 1 summarizes notations of BTN used in this paper. °, ⁿ and * denote sets of places, transitions and addresses. For example, In Fig. 3, $°t_0=\{p_6\}$, $t_0°=\{p_0\}$, $*t_0=\{a_4\}$, $t_0*=\{a_0\}$, $^{n}p_0=\{t_0\}$, $p_0^{n}=\{t_2\}$, $°a_0=\{p_0, p_2, p_4\}$, $^{n}a_0=\{t_0, t_2, t_3\}$ and $a_0^{n}=\{t_2\}$.

Table 1. Notations of BTN

No	Notation	Meaning
1	$°t=\{p\|(p, t)\in F\}$	The set of input places of transition t
2	$t°=\{p\|(t, p)\in F\}$	The set of output places of transition t
3	$°a=\{p\|\gamma(p)=a\}$	The set of places associated with the same address a
4	$^{n}p=\{t\|(t, p)\in F\}$	The set of input transitions of place p
5	$p^{n}=\{t\|(p, t)\in F\}$	The set of output transitions of place p
6	$^{n}a=\{t\|\gamma(p)=a\wedge t\in{}^{n}p\}$	The set of transitions whose output places are mapped to address a
7	$a^{n}=\{t\|\gamma(p)=a\wedge t\in p^{n}\}$	The set of transitions whose input places are mapped to address a
8	$*t=\{\gamma(p)\|p\in°t\}$	The set of addresses associated with all the input places of t
9	$t*=\{\gamma(p)\|p\in t°\}$	The set of addresses associated with all the output places of t.

V. Transaction Features

A. Static and Dynamic Features

This paper introduces 19 features to define transaction patterns. Other features can be added into our framework according to the requirement of analysis.

Table 2 presents the 19 features and their definitions including 6 new notations. First and last transaction time of address a is denoted by $\tau_f(a)$ and $\tau_l(a)$. *a.balance* and *a.received* denote coin balance and the total received coin amount of an address a. The Bitcoin gene is a novel feature proposed in our paper to analyze the Bitcoin flow. Its computations and evolvement will be introduced in the following subsection. Coins' gene of *a.balance* and *a.received* are represented by *a.balance.gene* and *a.received.gene*.

Features 1-14 are the static features which can be obtained from the structure of BTN. Features 15-17 are the dynamic features which can be analyzed by the firing of BTN, which will be introduced in the next section. Features 1-6 are the features related to the transaction (transition); Features 7-13, 16 and 17 are the address features; Features 14 and 15 are the place features; Features 18 and 19 are the gene features.

Table 2. 19 Features

No	Feature	Formal expression
1	Transaction time of transaction t	$\tau(t)$
2	Number of inputs of transaction t	$\|{}^\circ t\|$
3	Number of outputs of transaction t	$\|t^\circ\|$
4	Number of input addresses of transaction t	$\|{}^*t\|$
5	Number of output addresses transaction t	$\|t^*\|$
6	Transferred coin amount of transaction t	$\sum_{p\in t^\circ}\beta(p)$
7	Number of times that address a has occurred in all the transaction outputs	$\|{}^\circ a\|$
8	Number of deposit transactions of address a	$\|{}^\square a\|$
9	Number of withdrawing transactions of address a	$\|a^\square\|$
10	First transaction time of address a	$\tau_f(a)=\tau(t)$ such that $\tau(t)\le\tau(t_i)$ for $t, t_i\in{}^\square a\cup a^\square$
11	Last transaction time of address a	$\tau_l(a)=\tau(t)$ such that $\tau(t)\ge\tau(t_i)$ for $t, t_i\in{}^\square a\cup a^\square$
12	Number of incoming addresses that transfer coins to address a	$\|\{a'\|\gamma(p)=a \wedge t\in{}^\square p\wedge a'\in{}^*t\}\|$
13	Number of outgoing addresses to which coins are transferred from address a	$\|\{a'\|\gamma(p)=a \wedge t\in p^\square\wedge a'\in t^*\}\|$
14	Coin amount of transaction output p	$\beta(p)$
15	Whether an output p is spendable	$M(p)$
16	Coin balance of an address a	*a.balance*
17	Total received coin amount of an address a	*a.received*
18	Coins' gene of *a.balance*	*a.balance.gene*
19	Coins' gene of *a.received*	*a.received.gene*

B. Bitcoin Gene

The notion of "Bitcoin gene" is proposed to track the origination and distribution of balance (received) coins of a given address. This is analogous to a "gene" in biology, which is transferred from a parent to offspring and can be used to determine some characteristics of the offspring. We use a gene to indicate where the Bitcoins passed through a given address (called dyeing address) have flown and to determine the relationship strength between the dyeing address and other addresses. In the Bitcoin system, only addresses are related to users. There are two address features related to coins, which are *a.balance* and *a.received*. Therefore, *a.balance.gene* and *a.received.gene* are the two gene features used in our method.

Definition 2. A Bitcoin gene G is a set of DNAs, $\{(d_0, q_0, w_0), \ldots(d_n, q_n, w_n)\}$, where d_i, q_i, and w_i are the DNA name, Bitcoin quantity, and weight in DNA (d_i, q_i, w_i), respectively. d_i is either a dyeing address or a dummy name ϕ, $d_i\neq d_j$ $(i\neq j)$, and $\sum w_i=100\%$. $\sum q_i$ represents the quantity of Bitcoins with the gene G that we analyzed.

The gene definition in Definition 2 is called a row-definition of gene, which is inconvenient for gene operations. Therefore, the gene $\{(d_0, q_0, w_0), \ldots(d_n, q_n, w_n)\}$ is transferred into a matrix shown in Table 3 and a col-definition of gene is proposed in Definition 3.

Table 3. The Matrix of Bitcoin Gene

	D	$\delta(d_i)$	$\theta(d_i)$	
(	d_0	q_0	w_0	)
…	…	…	…	…
(	d_n	q_n	w_n	)

Definition 3. Given a Bitcoin gene $G=\{(d_0, q_0, w_0), \ldots(d_n, q_n, w_n)\}$, it can be defined as a 3-tuple $G=(D, \delta, \theta)$, where $D=\cup d_i$; δ is a function from D to positive integer N, $\delta(d_i)=q_i$; and θ is a function from D to (0,1], $\theta(d_i)=w_i$, $\sum d_i=1$.

Three gene operations, merging, splitting and dyeing are defined to evolve and propagate Bitcoin genes when a transaction occurs. Assuming $G_1=(D_1, \delta_1, \theta_1)$, $G_2=(D_2, \delta_2, \theta_2)$, $G_3=(D_3, \delta_3, \theta_3)$, their definitions are as follows.

Definition 4. Given G_1 and G_2, G_1 merging with G_2 is denoted as $G_1\perp G_2$. Assume $G_3=G_1\perp G_2$, then,

1. $D_3=D_1\cup D_2$;
2. $\delta_3=\{(d, y)|d\in D_3\wedge y=\delta_1(d)+\delta_2(d))\}$; (Note that ignore $\delta(d)$ if $d\notin D$.)
3. $\theta_3=\{(d, z)|d\in D_3\wedge z=\delta_3(d)/(\sum_{d'\in D_3}\delta_3(d'))\}$.

The merging operation is used to merge Bitcoin genes. Line 1 depicts the union of D_1 and D_2 as the DNA name set of the merged gene. Line 2 represents the Bitcoin quantity of each DNA in the merged gene as the sum of Bitcoin quantities with the same DNA in G_1 and G_2. Line 3 represents the percentages of each DNA in the merged gene as the quotient of Bitcoin quantity in a DNA divided by the total Bitcoin quantity contained in G_3.

Definition 5. Given G_1 and $n\in N^+$ ($n\le\sum_{d\in D_1}\delta_1(d)$), G_1 splitting with n is denoted as $G_1\backslash n$. Assume $G_2=G_1\backslash n$, then,

1. $D_2=D_1$;
2. $\delta_2=\{(d, y)|d\in G_2\wedge y=\theta_1(d)\times n\}$;
3. $\theta_2=\theta_1$;

4. If $n < \sum_{d \in D_1} \delta_1(d)$, $G_1=\{D_1, \{(d, y)|d \in G_1 \wedge y=\delta_1(d)-\theta_1(d) \times n\}, \theta_1\}$;
5. If $n = \sum_{d \in D_1} C_1(d)$, $G_1=\varnothing$.

The splitting operation is used to split gene homogeneously. Note that the splitting operation gives a result of G_2 and affects G_1 simultaneously. Line 1 sets the DNA name set of the split gene which is equal to the DNA name set of G_1. Line 2 sets Bitcoin quantity of each DNA in G_2 to $\theta_1(d) \times n$. Line 3 sets the percentage of each DNA in G_2 equal to the one in G_1. Line 4 and Line 5 define how the splitting operation affects G_1. Line 4 denotes that if the split Bitcoin quantity is less than the Bitcoin quantity of G_1, DNA name and its percentage of each DNA in G_1 maintain the same Bitcoin quantity of each DNA in G_1 and is set to $\delta_1(d)-\theta_1(d) \times n$. Line 5 is used to deal with a specific case, where if the split Bitcoin quantity equals to the Bitcoin quantity of G_1, namely all DNAs would be moved out from G_1, G_1 is empty.

Definition 6. Given G_1 and a DNA d, G_1 being dyed by d is denoted as $G_1 \leftarrow d$. Assume $G_2=G_1 \leftarrow d$, then,

1. $D_2=\{d\}$;
2. $\delta_3=\{(d, \sum_{d' \in D_1} \delta_1(d'))\}$;
3. $\theta_3=\{1\}$.

If an address a is a dyeing pool, it should have a dyeing DNA. All Bitcoins transferred to the address a should be dyed by the DNA. Line 1 changes the DNA name set to $\{d\}$. Line 2 sets the Bitcoin quantity of d to the Bitcoin quantity of G_1. Line 3 sets the percentage of d to 100%.

VI. Analysis of BTN

A. The initial state of BTN

The dynamic features are obtained through the firing of BTN transitions to simulate Bitcoin transaction occurrences.

Algorithm 1 below describes the way of transforming a Bitcoin Blockchain into a BTN, along with setting up its initial values.

Algorithm 1. Construction of BTN from a Bitcoin Blockchain.
Input: A block chain (a list of blocks).
Output: (P, T, F, A, β, γ, τ, M_0)

1. For each block
2. For each Bitcoin transaction t in the current block
3. $T=\{t\} \cup T$;
4. $\tau(t)$=the current block's timestamp;
5. For each output p of transaction t;
6. $P=\{p\} \cup P$;
7. $M_0(p)=0$;
8. $\beta(p)$=Bitcoin quantity locked in the output;
9. $F=\{(t, p)\} \cup F$;
10. $\gamma(p)$=address a extracted from the ScriptPubKey field of output p;
11. $A=\{a\} \cup A$;
12. $a.balance=0$;
13. $a.received=0$;
14. $a.balance.gene=\varnothing$;
15. $a.received.gene=\varnothing$;
16. End for;
17. For each input p of transaction t
18. $F=(p, t) \cup F$;
19. End for;
20. End for;
21. End for;
22. For each $t \in \{t| \circ t|=0\}$
23. Create a virtual and unique place p and $P=\{p\} \cup P$;
24. $M_0(p)=1$;
25. $F=\{(p, t)\} \cup F$;
26. $\beta(p)=\sum_{p' \in \circ t} \beta(p')$;
27. Generate a virtual and unique address a and $A=\{a\} \cup A$;
28. $\gamma(p)=a$;
29. $a.balance=\beta(p)$;
30. $a.received=\beta(p)$;
31. $a.balance.gene=\{(\phi, \beta(p), 100\%)\}$;
32. $a.received.gene=\{(\phi, \beta(p), 100\%)\}$;
33. End for

Lines 1-21 parse data from the Blockchain and construct a BTN. Line 3 adds a new transition to the transition set. Line 4 uses the block's timestamp as the transaction's timestamp since the Blockchain data does not save a transaction's timestamp. Lines 5-16 process the outputs of the transaction. Line 6 creates a new place for the output. Line 7 state that its usability is false, namely, it cannot be used as an input of other transactions. Line 8 sets up the value of $\beta(p)$, which can be obtained from the Blockchain data. Line 9 adds (t, p) to F. Line 10 sets up the mapped address of $\gamma(p)$, which can be obtained from the ScriptPubKey field of output in the Blockchain. Line 11 adds the address a to the address set A. Lines 12/13 and 14/15 set up the balance/received coins and balance/received gene to 0 and empty. Lines 17-19 add each input (p, t) to F.

Lines 22-33 add a virtual input and a virtual address to every coinbase transaction and configure the relative values. Line 23 creates a virtual and unique place/input and adds it to a place set. Line 24 sets up the input as usable, namely, it can be used as an input of a coinbase transaction. Line 25 adds (p, t) to F. Line 26 sets up the value of $\beta(p)$ to the sum coin quantities of outputs of the coinbase that uses the place as input. Line 27 generates a virtual and unique address and adds it to the address set. Line 28 sets up the map value of $\gamma(p)$ to the address. Lines 29/30 and 31/32 set up the balance/received coins and balance/received gene to the coin quantity of the input and $\{(\phi, \beta(p), 100\%)\}$. $\{(\phi, \beta(p), 100\%)\}$ as a row-definition of gene. Its col-definitions are $\{\{\phi\}, \{(\phi, a.balance)\}, \{(\phi, 100\%)\}\}$, where ϕ represents a background DNA. Before Bitcoins are mined, their genes usually stay pure and unpolluted. The pure genes contain only one DNA ϕ, which are usually ignored as they are not related to any suspected address.

By implementing Algorithm 1, a BTN is created, which can be fired to simulate transaction occurrences and calculate the features' values.

Before the BTN firing process, each dyeing DNA of every

address is setup to empty (dyeing DNA of an address being empty means the address is not a dyeing pool). If we want to trace the passing through of Bitcoins with specific address, we need to set the corresponding address as a dyeing pool. *a.dyeingDNA* denotes dyeing DNA of address *a*. *a.dyeingDNA* should be given a specific identifier or name assuming *d*, i.e., *a.dyeingDNA*=*d*. In our framework, addresses are pre-clustered based on the input address clustering method. Every dyeing DNA of each address in the same cluster with address *a* should also be setup to *d* because addresses within a cluster are very likely belong to a user or group.

B. Firing of BTN

If a transition *t* satisfies the following conditions, it can be fired.

Definition 7. Conditions of firing of transition *t*:

1. For $\forall p \in {}^\circ t$, $M(p)=1$;
2. $\sum_{p \in ot} \beta(p) \geq \sum_{p' \in to} \beta(p')$;
3. For $\forall a \in {}^*t$, $a.balance \geq \sum_{p \in ot \wedge \gamma(p)=a} \beta(p)$.

Condition 1 depicts that the inputs of transaction *t* are available. Condition 2 depicts that the Bitcoins locked in inputs of transaction *t* are more in quantity than Bitcoins locked in its corresponding outputs. Condition 3 states for each address of transaction *t*, its balance is no less than the Bitcoins withdrawn from the respective address.

Results of firing a transition *t* are as follows.

Definition 8. Effects of firing of transition *t*:

Assume that a temp gene $G=\varnothing$ in initial state. When *t* is firing,

For $\forall p \in {}^\circ t$,

1. $M(p)=0$.
2. $\gamma(p).balance=\gamma(p).balance-\beta(p)$;
3. $G=G \perp (\gamma(p).balance.gene \backslash \beta(p))$;

For $\forall p \in t^\circ$,

4. $M(p)=1$;
5. $\gamma(p).balance=\gamma(p).balance+\beta(p)$;
6. $\gamma(p).received=\gamma(p).received+\beta(p)$;
7. $G'=G \setminus \beta(p)$;
8. $\gamma(p).balance.gene=\gamma(p).balance.gene \perp G'$;
9. $\gamma(p).received.gene=\gamma(p).received.gene \perp G'$;
10. If $\gamma(p).dyeingDNA \neq \varnothing$, then $\gamma(p).balance.gene = \gamma(p).balance.gene \leftarrow \gamma(p).dyeingDNA$.

Lines 1-4 define the behaviors of transition's inputs. Line 1 means transition's firing consumes tokens from its input places. Line 2 means that the balance of the address mapped by the place *p* is subtracted by the Bitcoin quantity withdrawn by *p*. Line 3 means that the balance gene of the address mapped by *p* is split by $\beta(p)$; the split gene is merged to *G*.

Lines 4-9 define the behaviors of transition's outputs. Line 1 means that the transaction produces a new token for the place. Line 5/6 adds $\beta(p)$ to the place's balance/received coins. Line 7 splits $\beta(p)$ gene from *G* to *G*'. Line 8/9 merges *G*' to the balance/received gene of address mapped by *p*. Line 10 means that if the address mapped by *p* is a dyeing pool, its balance gene should be dyed by its dyeing DNA.

Firing sequence is a sequence of transitions that represents an occurrence order of Bitcoin transactions. The Bitcoin Blockchain itself maintains a transaction sequence. The BitcoinDatabaseGenerator[31], i.e., the open source tool we used to parse the Bitcoin Blockchain, does not restore the transaction sequence. (One of our further works is to improve this Blockchain parsing tool.) In order to recover the transition sequence as close as possible to the transaction occurrence sequence, we find a firing sequence according to the transaction timestamps. The following algorithm describes the process of firing of transitions.

Algorithm 2 Transition firing.

Input: an initialized BTN $N=(P, T, F, A, \beta, \gamma, M_0)$

Output: a BTN *N* with feature values.

1. $T'=\{t | M(p)=1 \wedge \forall p \in {}^\circ t\}$;
2. While $T' \neq \varnothing$ do
3. Find $t \in T'$ such that $\tau(t) \leq \tau(t_i)$ for $\forall t_i \in T' \wedge t \neq t_i$;
4. Fire *t* using Definition 8;
5. $T'=\{t | M(p)=1 \wedge \forall p \in {}^\circ t\}$;
6. End while

Line 1 is used to find a transition set, where every transition in the set can be fired. Line 2-6 are used to fire transitions according to their fireable conditions and timestamps. Line 3 is to find a fierable transition with the smallest timestamp. Line 4 is used to fire the transition according to Definition 8. Line 5 is used to update the fireable transition set.

After processing Algorithm 2, a BTN with feature values is obtained.

C. An Example

The BTN presented in Fig. 3 is used as an example to illustrate how the values of dynamic features change during the transitions firing process. Assume that we want to trace Bitcoins passing through address a_0. Let $a_0.dyeingDNA=d$. In Fig. 3, a_1 and a_2 are clustered into the same class. Other addresses are clustered into different classes separately. Therefore, only a_0 is set as a dyeing pool. Assume that a firing sequence is $t_0t_1t_2t_3$ and their states are $M_0M_1M_2M_3$ M_4. Balance and received coins of an address can be derived from its balance gene and received gene. Therefore, in this example, we focus on the analysis of gene.

In the initial state M_0, only a_4 and a_5 have coins. Their balance/received gene are $\{(\phi, 2, 100\%)\}/\{(\phi, 1, 100\%)\}$ and $\{(\phi, 2, 100\%)\}/\{(\phi, 1, 100\%)\}$. When t_0 is fired, 2 coins are moved from a_4 to a_0. Since a_0 is a dyeing pool, its balance and received gene become $\{(d, 2, 100\%)\}$ and $\{(d, 2, 100\%)\}$. The balance gene of a_4 becomes $\varnothing$. When t_1 is fired, 1 coin is moved from a_5 to a_1. The balance genes of a_5 and a_1 become $\varnothing$ and $\{(\phi, 1, 100\%)\}$. Received gene of a_1 becomes $\{(\phi, 1, 100\%)\}$. When t_2 is fired, 2 coins in a_0 are moved to a_0 (1 coin) and a_2 (1 coin). The balance and received genes of a_2 are $\{(d, 1, 100\%)\}$ and $\{(d, 1, 100\%)\}$. The balance and received genes of a_0 become $\{(d, 1, 100\%)\}$ and $\{(d, 3, 100\%)\}$. When t_3 is fired, the BTN reaches the final state M_4. Bitcoins in a_1 and a_2 are

mixed and moved to a_0 and a_3. The balance and received genes of a_0 become {(d, 2, 100%)} and {(d, 4, 100%)}. The balance and received genes of a_3 become {(ϕ, 0.5, 50%), (d, 0.5, 50%)} and {(ϕ, 0.5, 50%), (d, 0.5, 50%)}. *balance.gene* and *received.gene* of every address in each state are shown in Table 4 and Table 5. Genes in the two tables are represented by the form of Definition 2 which is easier for understanding.

TABLE 4. BALANCE GENE OF EVERY ADDRESS IN EACH STATE

	M_0	M_1	M_2	M_3	M_4
a_0	∅	{(d, 2, 100%)}	{(d, 2, 100%)}	{(d, 1, 100%)}	{(d, 2, 100%)}
a_1	∅	∅	{(ϕ, 1, 100%)}	{(ϕ, 1, 100%)}	∅
a_2	∅	∅	∅	{(d, 1, 100%)}	∅
a_3	∅	∅	∅	∅	{(ϕ, 0.5, 50%) (d, 0.5, 50%)}
a_4	{(ϕ, 2, 100%)}	∅	∅	∅	∅
a_5	{(ϕ, 1, 100%)}	{(ϕ, 1, 100%)}	∅	∅	∅

TABLE 5. RECEIVED GENE OF EVERY ADDRESS IN EACH STATE

	M_0	M_1	M_2	M_3	M_4
a_0	∅	{(d, 2, 100%)}	{(d, 2, 100%)}	{(d, 3, 100%)}	{(d, 4, 100%)}
a_1	∅	∅	{(ϕ, 1, 100%)}	{(ϕ, 1, 100%)}	{(ϕ, 1, 100%)}
a_2	∅	∅	∅	{(d, 1, 100%)}	{(d, 1, 100%)}
a_3	∅	∅	∅	∅	{(ϕ, 0.5, 50%) (d, 0.5, 50%)}
a_4	{(ϕ, 2, 100%)}	{(ϕ, 2, 100%)}	{(ϕ, 2, 100%)}	{(ϕ, 2, 100%)}	{(ϕ, 2, 100%)}
a_5	{(ϕ, 1, 100%)}	{(ϕ, 1, 100%)}	{(ϕ, 1, 100%)}	{(ϕ, 1, 100%)}	{(ϕ, 1, 100%)}

VII. PATTERN ANALYSIS

A. Constitution of Pattern

Pattern matching aims to find addresses that satisfy a given pattern. However, it is not easy to describe a complicated pattern directly. Therefore, we propose instructions on how to define a pattern by logical expressions.

Fig. 4 presents the constitution of a pattern. A pattern is defined by a set of properties. A property is a set of feature expressions. A feature expression can be defined as a logical expression over features according to the clues provided. In fact, the pattern, property and feature expression are all logical expressions over features. A feature expression is used to describe the character of a Bitcoin transaction feature. A property expression is used to describe an aspect of a pattern. A pattern expression describes a pattern. Generally, if we want to define a pattern, we should first analyze the aspects included in the pattern. Then for each aspect, a property should be defined by one or more feature expressions.

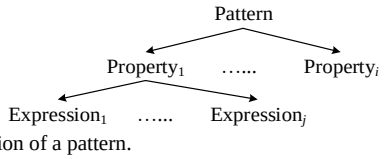


Fig. 4. Constitution of a pattern.

Different types of feature expressions can be classed into three levels. The first level is the address expression, denoted as E^1, which results in a set of addresses that satisfy E^1; the second level is the transaction expression, denoted as E^2, which results in a set of transactions that satisfy E^2; and the third level is the place expression, denoted as E^3, which results in a set of places that satisfy E^3.

A pattern results in a set of addresses since the aim of pattern matching is to find a set of addresses. A property also results in a set of addresses because a resulting address set of a pattern is the intersection of all the resulting address sets of the pattern's properties in our scheme. Therefore, a property expression must contain at least an address expression. Transaction expressions or place expressions cannot be used alone in a property expression. A transaction or place expression must be applied with an address expression combined by a joint clause.

A transaction feature expression E^2 can be used to refine an address expression E^1. A joint clause between E^1 and E^2 is $t\in{}^{\sqcap}a$ or $t\in a^{\sqcap}$. A place feature expression E^3 can be used to refine an address expression E^1. A joint clause between E^1 and E^3 is $\gamma(p)=a$. A place feature expression E^3 can be used to refine a transaction expression E^2. A joint clause between E^2 and E^3 is $p\in{}^{\circ}t$ or $p\in t^{\circ}$. Fig. 5 shows the refining relationships between E^1, E^2 and E^3. Table 6 presents the joint clauses between E^1, E^2 and E^3.

Address features, i.e. the level 1 features, are easy to use. For example, the address balance is used to find addresses whose balances are limited to a range. An expression $0<a.balance<100$ is used to find addresses whose balances are more than 0 but less than 100, which results in an address set $\{a|0<a.balance<100\}$. Different features can be combined to a complex expression. For example, an expression $(a.balance<100)\wedge(a.received>100)$ is used to find addresses where each of the balance is less than 100 and each of the total received coins are more than 100, which results in $\{a|(a.balance<100)\wedge(a.received>100)\}$.

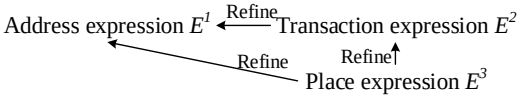


Fig. 5. Refining relationships between three level expressions.

TABLE 6. JOINT CLAUSES AND THEIR FUNCTIONS

Combination	E^1 and E^2	E^1 and E^3	E^2 and E^3
Clauses	$t\in{}^{\sqcap}a$, $t\in a^{\sqcap}$	$\gamma(p)=a$	$p\in{}^{\circ}t$, $p\in t^{\circ}$

A transaction feature (a Level-2 feature) expression needs a joint clause to become an address feature (level 1 feature) expression. For example, an express $(t\in{}^{\sqcap}a)\wedge(\sum_{p\in t^{\circ}}\beta(p)>10)$ is used to find addresses whose deposit transaction transfers more than 10 Bitcoins. $t\in{}^{\sqcap}a$ is a joint clause. The resulting address set is $\{a|(t\in{}^{\sqcap}a)\wedge(\sum_{p\in t^{\circ}}\beta(p)>10)\}$.

A place feature (a level-3 feature) expression needs a joint clause to become an address feature (a level-1 feature) expression. For example, an expression $(\gamma(p)=a)\wedge(M(p)=1)$ is used to find addresses having UTXOs. $\gamma(p)=a$ is joint clause. The resulting address set is $\{a|(\gamma(p)=a)\wedge(M(p)=1)\}$. An expression $(t\in{}^{\sqcap}a)\wedge(p\in t^{\circ})\wedge(\beta(p)>10)$ is used to find addresses where every output of each deposit transaction of the address a moves more than 10 Bitcoins to this address a. The resulting address set is $\{a|(t\in{}^{\sqcap}a)\wedge(p\in t^{\circ})\wedge(\beta(p)>10)\}$. This expression contains two joint clauses. $\beta(p)>10$ is a place feature

expression, which is combined with a joint clause $p \in t^{\circ}$. Then expression $(p \in t^{\circ}) \wedge (\beta(p) > 10)$ becomes a transaction feature expression. After that, $(p \in t^{\circ}) \wedge (\beta(p) > 10)$ is combined with joint clause $(t \in {}^{\sqcup}a)$, thus $(t \in {}^{\sqcup}a) \wedge (p \in t^{\circ}) \wedge (\beta(p) > 10)$ becomes an address expression.

A matching program of a pattern is written manually at present in our experiment. However, it would be more convenient to develop a compiler to compile patterns into programs automatically, which is a part of our further work.

B. Marginal Distribution Analysis of Feature Values

For a specific case, a set of addresses can be obtained once pattern matching is completed. However, it is possible that some addresses matching the rules may not relate to the case. These samples of addresses are false positive samples. Marginal distribution analysis is used to try to eliminate such false positive samples.

An assumption of the marginal distribution analysis method is that behaviors in a transaction pattern are similar. This assumption is derived from another assumption that a user's repeated behaviors in regards to a specific activity are similar. This assumption is derived from statistical facts that have been widely used in recommendation methods and abnormal behavior detection methods. Therefore, samples of a feature directly or indirectly related to addresses found by through pattern matching should distribute near similar values. The samples far away from the values are less likely to be relevant to the case. Therefore, these deviated samples can be eliminated to improve accuracy.

For example, let us assume an address set is obtained by pattern matching. Its value range of the total received coin amount is $[x, y]$. The range is divided into z intervals. If most samples (β%) are distributed into some continuous intervals, then (1-β)% samples can be deleted. The threshold β is an empiric value. A histogram can be used to assist with determining the value of β.

A subset can be obtained by removing the false negative samples based on their features. Different features can obtain different subsets. An intersection of these subsets delivers the final result set.

VIII. Case Study

A. The Mt. Gox Case

Mt.Gox [32] is a Bitcoin exchange launched in July 2010. In February 2014, Mt.Gox filed for bankruptcy protection since more than 500k Bitcoins have gone missing, which are to be likely stolen. Loss of Bitcoins probably started in August 2011 and lasted until late 2013. Based on the analysis of a Mt.Gox deposit and withdrawal log data, which is leaked in 2014 (no longer publicly available now), WizSec identified a transfer pattern, as shown in Fig. 6 [33]. In this pattern, some Bitcoins have been collected from different addresses and stored into larger holding addresses (also called gathering addresses). After that, these Bitcoins have been split and deposited onto different exchanges. Although WizSec has identified millions of possible Mt.Gox addresses, these addresses are not available. In this paper, we use the transfer pattern shown in Fig. 6 and the public Mt.Gox address "1LNWw6yCxkUmkhArb2Nf2MPw6vG7u5-WG7q" as the clues for further investigation. Our goal is to identify gathering addresses that match the transfer pattern. These addresses are likely to be used to transfer the lost Bitcoins.

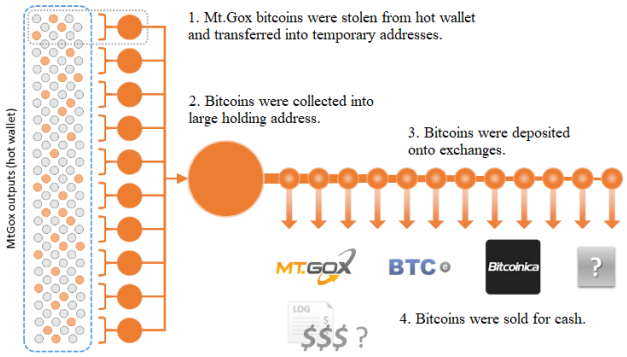


Fig. 6. A recurring transfer pattern of the lost Bitcoins form Mt.Gox [33].

B. Experiment Setup and Preprocessing

The selected time window of Bitcoin transactions is chosen from Jan/03/2009 to Dec/28/2013, which consists of more than 30,000,000 transactions. A virtual machine with 8 core CPU and 64G RAM is used for our analysis.

The transaction data is parsed from the Bitcoin Blockchain and stored in a SQL Server Database using BitcoinDatabase-Generator[31]. All addresses are clustered by the input address clustering method [8, 10-14]. The clustered information is stored in the SQL Server Database. Mt.Gox address "1LNWw6yCxkUmkhArb2Nf2MPw6vG7u5WG7q" is clustered into a class that contains 544k addresses. All these addresses in the class are setup as dyeing pools. The dyeing gene of each of them is setup to *mgDNA*.

Creating BTN and its firing process are achieved in less than 20 minutes and less than 40 minutes respectively. On the contrary, the pattern analysis takes less than a minute, which thus the entire process is completed in an hour.

C. Pattern Setup

WizSec pointed that "Mt.Gox Bitcoins have been sent to a new non-Mt.Gox address often in fairly recognizable amounts of a few hundred BTC at a time. Shortly afterwards, these addresses in turn would be gathered up into bigger addresses holding a few thousand BTC. From there, the coins would get deposited in chunks of some hundred BTC at a time onto various Bitcoin exchanges." The transfer pattern is shown in Fig. 6. The large holding addresses, also called gathering addresses, are used as distinguishable points.

According to the clues indicated by WizSec, we formalized the transfer pattern with the following 7 properties in which a represents a gathering address.

1. The number of incoming addresses of a gathering address a is not less than 3. The property is $|\{a' | \gamma(p)=a \wedge t \in {}^{\sqcup}p \wedge a' \in {}^{*}t\}| \geq 3$.
2. "Each gathering address holds a few thousand Bitcoins". According to this, a total amount of received Bitcoin of

a large holding address a is setup to 1000-10000. The property is $1000 \leq a.received \leq 10000$.

3. "The losing behavior started from August 2011". For the sake of safety, we set the start date to July 15, 2011. The property $(t \in {}^{\sqcap}a) \wedge (\tau(t) \geq \text{July/15/2011})$. $t \in {}^{\sqcap}a$ is a joint clause.
4. Every transaction that transferred Bitcoins to a gathering address a has only one output place. The property $(t \in {}^{\sqcap}a) \wedge (|t^{\circ}|=1)$. $t \in {}^{\sqcap}a$ is a joint clause.
5. There are two methods to accept coin change: using the gathering address, and using a new change address. It is unknown which method has been used in the Mt.Gox case. 1) If the first was used, there would be many outgoing addresses, namely, the number of outgoing addresses would be greater than 2; 2) If the second was used, every transaction that transferred Bitcoins from a gathering address would have two outputs belonging to two different new addresses. Thus, the number of outgoing addresses of a gathering address would be equal to 2. According to 1) and 2), the number of outgoing addresses of a gathering address a is no less than 2. The property is $(|\{a' | \gamma(p) = a \wedge t \in p^{\sqcap} \wedge a' \in t^*\}| \geq 2)$.
6. Every transaction that withdrew Bitcoins from a gathering address has only one incoming address. This property is $(t \in a^{\sqcap}) \wedge (^*t=1)$.
7. Received gene of every gathering address should contain *mgDNA*. This property is $mgDNA \in a.receive\text{-}d.gene.D$.

The above 7 properties defined the pattern. Through analysis, an address set denoted as S containing 236 addresses is found, which transferred 560K bitcoins.

D. Marginal Distribution Analysis

The 236 addresses found are likely to contain false positive samples, namely some found addresses may follow the pattern but do not relate the Mt.Gox case. While the Marginal Distribution Analysis (MDA) is primarily attempted to spot and remove the false positive samples, it also found latent details hidden in the pattern.

For example, as mentioned previously in section VIII.C, there are two methods to accept coin change, the pattern does not give insights into the method that is actually used. The marginal distributions in the number of withdrawn transactions and the number of outgoing addresses can reveal such details. Fig. 7 and Fig. 8 present the two distributions. Only 16 gathering addresses from the totally found 236 gathering addresses characterize more than 2 withdraw transactions and outgoing addresses, which means most of them have used new addresses to accept changes. This finding also conforms with the fact that using a new address to receive change is safer than reusing an old address. Then an address subset by excluding the 16 addresses is obtained, denoted as S_1.

When Bitcoins have been collected to a gathering address, information on the number of involved transactions collecting these Bitcoins are not known. Distribution of the number of deposit transactions of gathering addresses, as shown in Fig. 9, reveal such details. There are only 23 addresses characterizing 2 or more deposit transactions, which means most of them have actually used just a single transaction to collect Bitcoins. This conforms with the fact that fewer number of transactions cost less transaction fee. Then an address subset excluding these 23 addresses is obtained, denoted as S_2.

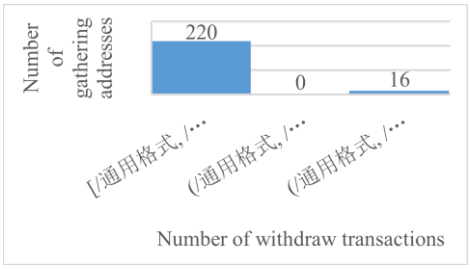


Fig. 7. Distribution of the number of withdrawn transactions of gathering addresses.

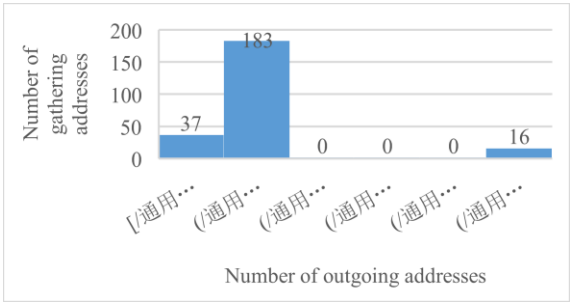


Fig. 8. Distribution of the number of outgoing addresses of gathering addresses.

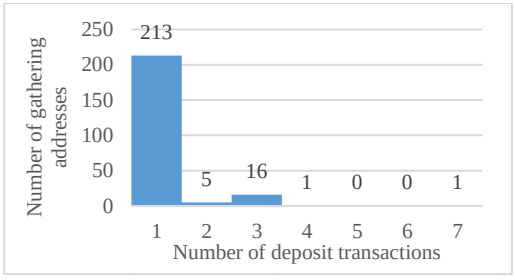


Fig. 9. Distribution of the number of deposit transactions of gathering addresses.

Fig. 10 presents the relationship between S, S_1 and S_2. About 90% (213) addresses share the same characteristics.

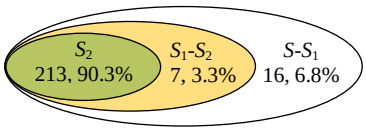


Fig. 10. Relationships between S, S_1 and S_2.

For some features, it is not difficult to determine their thresholds which decide whether to include or exclude related addresses. However, for other features, their appropriate threshold values are not obvious. This entirely depends on the analysis requirements. If a high precision is required, a radical threshold should be selected to remove more addresses. If a high recall is needed, a conservative threshold should be selected to retain more addresses. For the above three features, it is not difficult to determine their thresholds. However, the following two features is on the contrary.

WizSec stated that Mt.Gox Bitcoins have been collected in

amounts of a few hundred Bitcoins to form the gathering addresses which then comprised a few thousand Bitcoins. Nevertheless, we do not know the number of addresses collected to form the gathering addresses. Fig. 11 presents the distribution of the number of incoming addresses of gathering addresses, where 12, 18 and 51 are the optional thresholds. If we want to obtain addresses that have much higher similarity, 12 is a reasonable selection. If we focus on the recall of the related addresses, 51 is an alternative. Herein, 18 is selected as a threshold. The obtained address subset, denoted as S_3, contains 203 addresses.

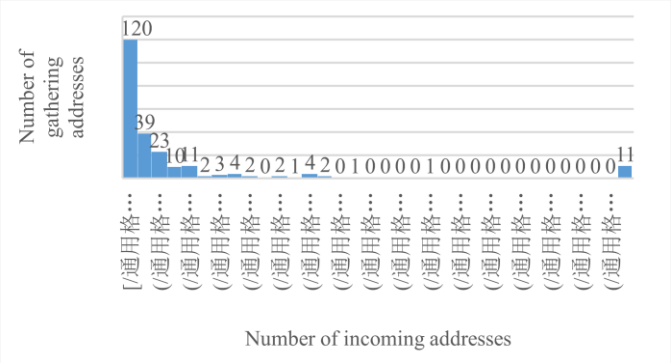


Fig. 11. Distribution of the number of incoming addresses of gathering addresses.

In some cases, we need to combine features to form new features for analysis. For example, the pattern information does not provide the information regarding the duration required to split the gathering Bitcoins into exchanges. Even among the 19 features, none of the features has represented the time span directly. However, there are two features, the first and the last transaction time of addresses, which can be used to obtain the time span feature. Fig. 12 presents the distribution of time span between depositing and withdrawing Bitcoins to or from gathering addresses. There are many different thresholds, which can be selected, such as 1, 4 and 7 according to requirements of precision and recall. Herein, 4 is selected as a threshold. An address subset S_4 is obtained, which contains 186 addresses.

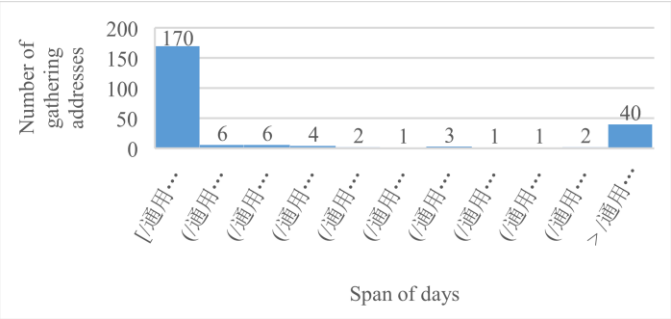


Fig. 12. Distribution of time span of depositing and withdrawing bitcoins to and from gathering addresses.

Finally, an intersection of S_1, S_2, S_3 and S_4 is obtained. It contains 161 addresses that transferred about 352K bitcoins. The addresses in this intersection share similar features, and very likely they belong to the same person or group. The purpose of our analysis is to provide the 161 addresses that are very likely related to the pattern. Then law enforcement agencies can trace these Bitcoins. If they are transferred into an exchange and the exchange has the identity information, then suspects can be found effectively. It is not appropriate to argue that all of the identified addresses relate to the required pattern. But the results do provide very useful information for the analysis of lost Bitcoins.

The Mt.Gox case is used as an example to illustrate the effectiveness and efficiency of the proposed method in analyzing Bitcoin patterns. From the above analysis, we can see that the proposed method is very flexible and can be used for various and complicated cases analysis.

IX. Conclusions

This paper proposed a novel framework for Bitcoin transaction network analysis. In this framework, Bitcoin transactions are formalized as an extended Safe Petri net, called BTN. Its structure and semantic features are used to describe Bitcoin transaction's static and dynamic features. The gene feature of Bitcoins can be used for the Bitcoin flow analysis. Based on the described features, various transaction patterns can be defined. The addresses matched with the patterns can be identified. The proposed method has been proven to be an efficient tool for future Bitcoin transaction forensic investigation, based on real life case study analysis.

In our experiments, pattern expressions are programmed manually. In the next step, a compiler will be developed to compile patterns into programs automatically. Bitcoin Blockchain data is tremendous, if intermediate states are saved at different time points, a considerable amount of time can be saved during a recent case analysis. Next, we will continue to investigate the means of preserving intermediate states of BTN. In our experiments, an open source tool, BitcoinDatabase-Generator, is used to parse Bitcoin Blockchain. However, it does not recover block and transaction orders information which affects the analysis performance. Investigating this drawback of this tool is our another research objective as future work. An integrated bitcoin analysis platform is in our future plan to assist with bitcoin forensic analysis.

Acknowledgment

This work was partially supported by the Natural Science Foundation of Jiangsu Province under Grant BK20170069, the National Natural Science Funds of China under Grants No. U1836116, the National Key Research and Development Program of China under Grants No. 2017YFB1400703, the Postdoc Funds of China and Jiangsu Province under Grants No. 2015M580396 and 1501023A. UK-Jiangsu 20-20 World Class University Initiative programme, and UK-Jiangsu 20-20 Initiative Pump Priming Grant.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[2] D. Bryans, "Bitcoin and money laundering: mining for an effective solution," *Indiana Law Journal*, vol. 89, pp. 1-33, 2014.

[3] M. J. Barratt, "SILK ROAD: EBAY FOR DRUGS: The journal publishes both invited and unsolicited letters," *Addiction*, vol. 107, pp. 683-683, 2012.

[4] M. Dittus, J. Wright, and M. Graham, "Platform Criminalism: The'last-mile'geography of the darknet market supply chain," in proceedings of *the 2018 World Wide Web Conference on World Wide Web*, 2018, pp. 277-286.

[5] G. White. *UK company linked to laundered Bitcoin billions*, BBC, (2018). Available: https://www.bbc.com/news/technology-43291026

[6] N. J. Ajello, "Fitting a Square Peg in a Round Hole: Bitcoin, Money Laundering, and the Fifth Amendment Privilege Against Self-Incrimination," *Brooklyn Law Review*, vol. 80, p. 4, 2015.

[7] P. Reynolds and A. S.M. Irwin, "Tracking digital footprints: anonymity within the bitcoin system," *Journal of Money Laundering Control*, vol. 20, pp. 172-189, 2017.

[8] F. Reid and M. Harrigan, "An Analysis of Anonymity in the Bitcoin System," in proceedings of *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, 2011, pp. 1318-1326.

[9] S. Göbel, *A Polynomial Translation of Mobile Ambients Into Safe Petri Nets: Understanding a Calculus of Hierarchical Protection Domains*: Springer, 2016.

[10] D. Ron and A. Shamir, "Quantitative Analysis of the Full Bitcoin Transaction Graph," in proceedings of *International Conference on Financial Cryptography and Data Security* Berlin, Heidelberg, 2013, pp. 6-24.

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

[12] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, and S. Capkun, "Evaluating User Privacy in Bitcoin," in proceedings of *International Conference on Financial Cryptography and Data Security*, Berlin, Heidelberg, 2013, pp. 34-51.

[13] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, *et al.*, "A fistful of bitcoins: characterizing payments among men with no names," presented at the Proceedings of the 2013 conference on Internet measurement conference, Barcelona, Spain, 2013.

[14] M. Spagnuolo, F. Maggi, and S. Zanero, "BitIodine: Extracting Intelligence from the Bitcoin Network," in proceedings of *International Conference on Financial Cryptography and Data Security*, Berlin, Heidelberg, 2014, pp. 457-468.

[15] M. Harrigan and C. Fretter, "The unreasonable effectiveness of address clustering," in proceedings of *2016 Intl IEEE Conferences 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)*, 2016, pp. 368-373.

[16] C. Zhao and Y. Guan, "A GRAPH-BASED INVESTIGATION OF BITCOIN TRANSACTIONS," in proceedings of *Advances in Digital Forensics XI*, Cham, 2015, pp. 79-95.

[17] D. D. F. Maesa, A. Marino, and L. Ricci, "Uncovering the Bitcoin Blockchain: An Analysis of the Full Users Graph," in proceedings of *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2016, pp. 537-546.

[18] D. Di Francesco Maesa, A. Marino, and L. Ricci, "Data-driven analysis of Bitcoin properties: exploiting the users graph," *International Journal of Data Science and Analytics*, September 25 2017.

[19] M. Ober, S. Katzenbeisser, and K. Hamacher, "Structure and anonymity of the bitcoin transaction graph," *Future internet*, vol. 5, pp. 237-250, 2013.

[20] A. Pinna, R. Tonelli, M. Orrú, and M. Marchesi, "A Petri Nets Model for Blockchain Analysis," *arXiv preprint arXiv:1709.07790*, 2017.

[21] A. Pinna, "A Petri Net-based Model for Investigating Disposable Addresses in Bitcoin System," in proceedings of *Knowledge Discovery on the WEB*, 2016, pp. 1-4.

[22] J. V. Monaco, "Identifying Bitcoin users by transaction behavior," in proceedings of *SPIE Defense + Security*, 2015, p. 15.

[23] M. A. Harlev, H. Sun Yin, K. C. Langenheldt, R. Mukkamala, and R. Vatrapu, "Breaking Bad: De-Anonymising Entity Types on the Bitcoin Blockchain Using Supervised Machine Learning," in proceedings of *the 51st Hawaii International Conference on System Sciences*, 2018, pp. 1-10.

[24] M. Möser, R. Böhme, and D. Breuker, "An inquiry into money laundering tools in the Bitcoin ecosystem," in proceedings of *APWG eCrime Researchers Summit*, 2013, pp. 1-14.

[25] G. D. Battista, V. D. Donato, M. Patrignani, M. Pizzonia, V. Roselli, and R. Tamassia, "Bitconeview: visualization of flows in the bitcoin transaction graph," in proceedings of *IEEE Symposium on Visualization for Cyber Security (VizSec)*, 2015, pp. 1-8.

[26] N. Christin, "Traveling the silk road: a measurement analysis of a large anonymous online marketplace," presented at the Proceedings of the 22nd international conference on World Wide Web, Rio de Janeiro, Brazil, 2013.

[27] D. Kondor, M. Pósfai, I. Csabai, and G. Vattay, "Do the rich get richer? An empirical analysis of the Bitcoin transaction network," *PloS one*, vol. 9, pp. 1-10, 2014.

[28] D. McGinn, D. Birch, D. Akroyd, M. Molina-Solana, Y. Guo, and W. J. Knottenbelt, "Visualizing dynamic bitcoin transaction patterns," *Big data*, vol. 4, pp. 109-119, 2016.

[29] S. Bistarelli and F. Santini, "Go with the -Bitcoin- Flow, with Visual Analytics," presented at the Proceedings of the 12th International Conference on Availability, Reliability and Security, Reggio Calabria, Italy, 2017.

[30] Y. Wu, A. Luo, and D. Xu, "Forensic Analysis of Bitcoin Transactions," in proceedings of *2019 IEEE International Conference on Intelligence and Security Informatics (ISI)*, 2019, pp. 167-169.

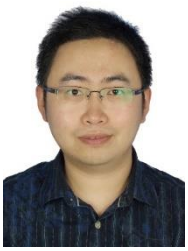
[31] ladimolnar. *BitcoinDatabaseGenerator*, (2017). Available: https://github.com/ladimolnar/BitcoinDatabaseGenerator

[32] Wikipedia. *Mt.Gox*, (2019). Available: https://en.wikipedia.org/wiki/Mt._Gox

[33] WizSec. *The missing MtGox bitcoins*, (2015). Available: https://blog.wizsec.jp/2015/04/the-missing-mtgox-bitcoins.html

Yan Wu is a Lecturer in the School of Computer Science and Telecommunication Engineering at Jiangsu University, China. He received the PhD degree from Tongji University, Shanghai, China, in 2014. He His research interests include blockchain, formal methods, service-oriented Computing, and big data.

Fang Tao received her PhD degree from Loughborough University, UK, in 2017 and MSc degree from Brunel University, UK, in 2005. She is a researcher at Jiangsu University and a visiting scholar at the University of Leicester, UK. Her research interests are in business analytics, data analytics and international business.



Lu Liu is the Head of School of Informatics and Professor of Informatics at the University of Leicester, UK. Prof. Liu received his Ph.D. degree from University of Surrey in 2008 and M.S. degree from Brunel University in 2003. Prof. Liu's research interests are in the areas of data analytics, service computing, cloud computing, Artificial Intelligence and the Internet of Things. He is a Fellow of British Computer Society (BCS).

Jiayan Gu received the BSc degree from University of Derby, UK, in 2018. She is currently working towards the PhD degree in the University of Derby, UK. Her research interests include service computing, edge computing and cloud computing. She is a Member of IEEE.

John Panneerselvam is a Lecturer in Computing at the University of Derby, United Kingdom. John received his PhD in Computing from the University of Derby in 2018 and an MSc in advanced computer networks in 2013. He is an active member of IEEE and British Computer Society, and a HEA fellow. His research interests include cloud computing, fog computing, Internet of Things, big data analytics, opportunistic networking and P2P computing. He has won the best paper award in IEEE International Conference on Data Science and Systems, Exeter, 2018.

Rongbo Zhu is currently a Professor in the College of Computer Science at South-Central University for Nationalities, China. Prof. Zhu received the B.S. and M.S. degrees in Electronic and Information Engineering from Wuhan University of Technology, China, in 2000 and 2003, respectively; and Ph.D. degree in communication and information systems from Shanghai Jiao Tong University, China, in 2006. His research interests include mobile computing, protocol design and performance optimization in wireless networks.

Mohammad Shahzad has done his Bachelor in Electronics and Master in Electrical Engineering from COMSATS University, Pakistan. He is currently doing PhD in Informatics at the University of Leicester. His research interests include Cloud Computing, IoT, Future Networks and Machine Learning.