Forensic Science International: Digital Investigation 31 (2019) 200895

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

journal homepage: www.elsevier.com/locate/fsidi

Identifying suspicious addresses in Bitcoin thefts

Yan Wu [a], Anthony Luo [b], Dianxiang Xu [c, *]

[a] School of Computer Science, Jiangsu University, Zhenjiang, Jiangsu, 212013, China
[b] Fu Foundation School of Engineering and Applied Science, Columbia University, New York, NY, 10027, USA
[c] Department of Computer Science and Electrical Engineering, University of Missouri – Kansas City, Kansas City, MO, 64110, USA

ARTICLE INFO

Article history:
Received 24 May 2019
Received in revised form
7 November 2019
Accepted 10 November 2019
Available online xxx

Keywords:
Blockchain
Bitcoin
Forensic analysis
Pattern matching

ABSTRACT

Bitcoin as a popular digital currency has been a target of theft and other illegal activities. Key to the forensic investigation is to identify bitcoin addresses involved in the bitcoin transfers. This paper presents a framework, FABT, for forensic analysis of bitcoin transactions by identifying suspicious bitcoin addresses. It formalizes the clues of a given case as transaction patterns defined over a comprehensive set of features. FABT converts the bitcoin transaction data into a formal model, called Bitcoin Transaction Net (BTN). The traverse of all bitcoin transactions in the order of their occurrences is captured by the firing sequence of all transitions in the BTN. When analyzing transaction flows, FABT exploits the notion of "bitcoin fluid" to track where the bitcoins passed through given addresses (called dyeing addresses) have flown and determine the extent to which each of the other addresses is related to the dyeing addresses. The splitting, merging, and dyeing operators are used to capture the distribution of coins throughout transaction flows. FABT also applies visualization techniques for further analysis of the suspicious addresses. We have applied FABT to identify suspicious addresses in the Mt.Gox case. A subgroup of the suspicious addresses has been found to share many characteristics about the received/transferred amount, number of transactions, and time intervals.

1. Introduction

Bitcoin (Nakamoto, 2008) has become increasingly popular as an electronic form of currency. Users hold bitcoins via addresses that are not linked to personally identifiable information. Therefore, bitcoins had been commonly used for trades at major darknet markets such as Silk Road, AlphaBay, and Hansa.[1] There have been numerous high-profile cases of bitcoin theft in the past. An operator of the BTC-e bitcoin exchange laundered more than $4 billion worth of illegal funds for criminals, ranging from computer hackers to drug traffickers. In 2014, the Mt. Gox exchange announced that approximately 850,000 bitcoins, valued at more than $450 million at the time, were missing and likely stolen. In 2015, 19,000 bitcoins, worth about $5 million at the time, were stolen from the Bitstamp exchange and in 2016, nearly 120,000 bitcoins, worth about $72 million at that time, were stolen from the Bitfinex exchange. NiceHash, a marketplace for mining digital currencies, announced in 2017 that 4700 bitcoins, worth $75 million at that time, were stolen from its account.

A key to the forensic investigation of such cases is the identification of bitcoin addresses that are involved in the related transactions. Although bitcoin holders are pseudonym, all transactions on the bitcoin blockchain are public. If a criminal bitcoin address is known, we can track the bitcoins that have passed through the address. If these coins are then deposited in Bitcoin exchanges (places that convert bitcoins to government-issued currencies), law enforcement would be able to obtain the suspect's identity information because Bitcoin exchanges are required by "Know Your Customer" laws to collect personal information.

This paper presents a framework, FABT, for forensic analysis of bitcoin transactions by identifying suspicious bitcoin addresses involved in a case under investigation. It formalizes the clues of a given case as transaction patterns defined over a comprehensive set of features regarding transactions, addresses, and transaction flows. To facilitate pattern matching, FABT converts the bitcoin transaction data into a formal model, called Bitcoin Transaction Net (BTN), which is an extended form of safe Petri nets (Göbel, 2016). The formal model allows the transaction and address features to be formalized in terms of the structural information of the BTN and

* Corresponding author.
E-mail addresses: wuyan04418@ujs.edu.cn (Y. Wu), anthony.luo@columbia.edu (A. Luo), dxu@umkc.edu (D. Xu).

[1] Such darknet sites as Silk Road, AlphaBay, and Hansa are no longer available.

https://doi.org/10.1016/j.fsidi.2019.200895
1742-2876/

the transaction flow features to be analyzed by the dynamic semantics of transition firing of the BTN. The traverse of all bitcoin transactions in the order of their occurrence is captured by the firing sequence of all transitions in the BTN. When analyzing transaction flows, FABT uses the notion of "bitcoin fluid" to track where the bitcoins passed through given addresses (called dyeing addresses) have flown and determine the extent to which each of other addresses is related to the dyeing address. The splitting, merging, and dyeing operators are used to capture the distribution of coins throughout transaction flows. In addition, FABT applies visualization techniques for further analysis of the suspicious addresses identified by the pattern matching.

We have applied FABT to the investigation of the Mt.Gox case according to the transfer pattern reported by WizSec (2015). The clues in the transfer pattern are formalized as a set of rules with respect to the features of transactions, addresses, and transaction flows. Our analysis resulted in 187 suspected gathering addresses. The visualizations of these addresses have also revealed that a subgroup of 16 addresses share many characteristics about the received/transferred amount, number of transactions, and time interval. Although these addresses are believed to be highly suspicious, verification of these addresses is beyond the scope of this paper.

The rest of this paper is organized as follows. Section 2 introduces related work. Section 3 presents the proposed framework. Section 4 describes formal modeling of bitcoin transactions. Section 5 focuses on analysis of bitcoin transactions. Section 6 presents the Mt.Gox case study. Section 7 concludes this paper.

2. Related work

Reid and Harrigan (2011) conducted temporal flow analysis, egocentric analysis and visualization of Bitcoin transactions based on the construction of a transaction network and a user network. Entities in the transaction network were formed by the clustering of multiple addresses via transactions with multiple inputs. External information from web sources was incorporated into the user network. Their method can track coins from a specified address and cluster addresses, but cannot indicate the strength of relationship between addresses. Tracking coins and calculating the strength of relationship between addresses are two functions of our proposed bitcoin fluid method. D. Ron and A. Shamir (Ron and Shamir, 2013) used the same clustering method to create a "contracted transaction graph". The statistical analysis revealed certain patterns and behaviors of large transactions that were possibly indicative of attempts to mask linkage. Their study focus on finding typical bitcoin spending/moving behavior features of bitcoin users by analyzing blockchain data, whereas our method is to find suspected addresses according to a defined transaction pattern. Fleder et al. (2015) also applied the clustering method with external information to tag entities and performed graph analysis. Their improvement is using PageRank to identify node importance. They do not address the question of finding suspected addresses by known information. Meiklejohn et al. (2013) used an additional clustering algorithm that clustered "change addresses". These addresses are created to collect changes when Bitcoins are sent during a transaction. As the "change address" is one of the transaction outputs, the initial address and the "change address" can be clustered together. BitIodine (Spagnuolo et al., 2014) is a framework for forensics analysis of Bitcoins, based on existing clustering heuristics in (Reid and Harrigan, 2011; Meiklejohn et al., 2013; Androulaki et al., 2013) in conjunction with data scraped from the web to create transaction and user graphs. It has been used to imply ownership of an address to the Silk Road and to find connections between Dread Pirate Roberts and an address. These address clustering methods can help to identity suspicious addresses. If address *a* is a known address that used to transfer stolen bitcoins, then the addresses in the same cluster with address *a* are suspicious addresses because they may belong to one user or group. However, address cluster methods only use transaction inputs and outputs information. Androulaki et al. (2013) demonstrated the effectiveness of behavioral analysis in blockchain transactions. Our method proposed 19 transaction features to defined transaction patterns, which can utilize various information of users' transaction behaviors.

Pinna et al. (2017) proposed an approach to Bitcoin analysis by creating Petri nets of Bitcoin addresses and address clusters. It focuses on the clustering of addresses that belong to a certain group or user with the use of external information to tag clustered addresses. It does not aim to find suspected addresses. In comparison, our paper exploits Petri nets as a formal model of bitcoin transactions in order to track bitcoin flow through specific addresses and find addresses of interest. Monaco (2015) proposed several transaction features, including hour of day, coin flow and input/output balance, to de-anonymize users from their transaction behavior over time. Harlev et al. (2018) used supervised machine learning to predict bitcoin cluster categorization. Our paper uses a more comprehensive set of transaction and address features for analysis.

Visualization methods have also been used for Bitcoin analysis. Moser et al. (Möser et al., 2013) utilized Bitcoin taint analysis and visualization to gauge the effectiveness of various mixing services. McGinn et al. (2016) focused on the use of large-scale transaction visualization and demonstrated patterns of money laundering, DDOS attacks, and potential application to the detection of transaction patterns such as tumbling and payment services. Battista et al. (2015) developed BitConeView, a bitcoin transaction visualization tool, and demonstrated a use case scenario to track Bitcoin money laundering in the experiments performed by Moser et al. (Möser et al., 2013). Bistarelli et al. (Bistarelli and Santini, 2017) created BlockChainVis, a tool for Bitcoin flow visualization with different filters and views to allow a user to capture visually interesting characteristics. Kondor et al. (2014) and Maesa et al. (Di Francesco Maesa et al., 2018) performed analysis of the topology of the Bitcoin network overall. Christin (2013) analyzed overall trends of users with regards to connections with illegal activity such as the Silk Road. Feder et al. (2017) analyzed the impact of security shocks (focusing on Mt.Gox in particular) on Bitcoin trade. Ron and Shamir (2014) tracked the flow of Bitcoins of Dread Pirate Roberts who ran the Silk Road. In our paper, visualization method is used as a tool for refining the formulation of bitcoin transaction patterns.

3. Overview of FABT

The problem of forensic analysis in FABT is formulated as follows: given the Bitcoin blockchain data and a set of clues of the case under investigation, including (input) bitcoin addresses (e.g., at which bitcoins were stolen or money laundering was started), we want to identify a set of (output) bitcoin addresses that likely held or still hold the bitcoins originally from the given input addresses. Similar to bank account numbers, addresses can be used to try to identify bitcoin owners. However, de-anonymization to find the owners of the suspicious bitcoin addresses is beyond the scope of this paper.

Fig. 1 presents the investigation workflow in our approach. First, we transform the bitcoin transactions in the given blockchain into a formal model, called bitcoin transaction net (BTN). As an extended form of safe Petri nets with well-defined semantics, BTN facilitates rigorous analysis of bitcoin transactions. We express clues as transaction patterns and extract information about the features involved in the transaction patterns by analyzing the bitcoin

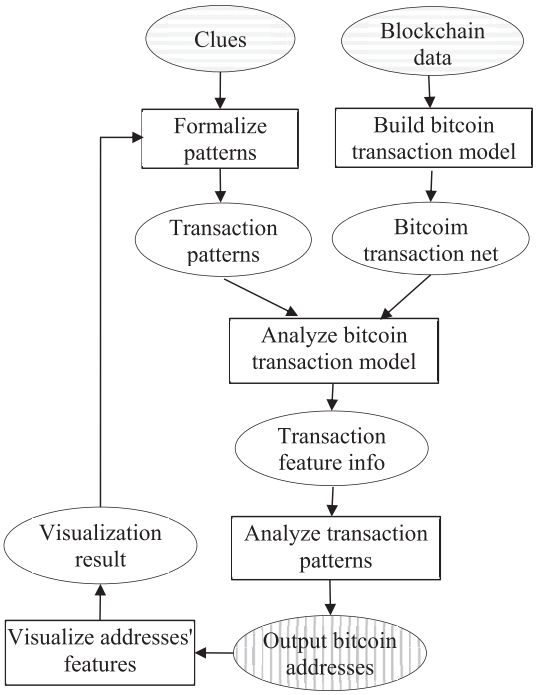


Fig. 1. The workflow of forensics investigation.

transaction net. This information is used to match the formalized transaction patterns to identify suspicious addresses. By visualizing the features of the resultant addresses, we can further refine the transaction patterns to improve accuracy.

Our approach allows transactions patterns to be defined over a comprehensive set of features with respect to bitcoin transactions and addresses. Transaction features include: transaction time, transferred coin amount, input transaction count, output transaction count, input address count, and output address count. These features, although straightforward, are useful for narrowing down the search space. For example, forensic cases of bitcoin transactions often involve transactions that gather or split coins. A transaction that gathers coins usually has a number of input transactions and one output transaction. A transaction that splits coins usually has one input address and a number of output addresses.

Address features include bitcoin balance, total received coins, first transaction time, last transaction time, number of deposit transactions, number of withdraw transactions, number of incoming addresses that transfer coins to the given address, number of outgoing addresses to which bitcoins are transferred from the given address, number of times that the address has occurred in the transaction outputs, distribution of balance (or total received) coins over the source addresses. A source address for a given address is one from which coins are transferred directly or indirectly to the given address.

4. Formal modeling of Bitcoin transactions

In this section, we first give a brief introduction to the structure of bitcoin transactions. Then we present how to transform bitcoin transactions into a bitcoin transaction net.

4.1. Bitcoin transactions

Bitcoin transactions are stored in a growing chain of blocks. Each block contains a number of transactions and each block is linked to its previous block by its hash value. Bitcoin miners group transactions into blocks according to transaction time and fees. As shown in Fig. 2, a bitcoin transaction consists of version, input count, a list of inputs (denoted as *input[]*), output count, a list of outputs (denoted as *outputs[]*), and lock time. Lock time refers to the point in time after which the transaction can be included in a block. Input count represents the number of inputs (i.e., the size of *input[]*). Each input, *input[i]*, consists of previous transaction hash, previous output index, script length, *ScriptSig*, and sequence. Previous transaction hash is obtained by applying SHA256 twice to the previous transaction. Previous output index references the output of the previous transaction that is used as the input of the current transaction. Script length specifies the size of *ScriptSig*, which is a script that unlocks the bitcoins in the corresponding output of the previous transaction. Sequence was originally intended for multiple signers to agree to update a transaction before including it in a block. Output count represents the number of outputs (i.e., the size of *output[]*). Each output, *output[j]*, consists of value, script length, and ScriptPubKey. Value is the quantity of bitcoins (unit is satoshi) locked in *output[j]*. Script length specifies the length of ScriptPubKey, which is a script for locking the bitcoins in *output[j]*. ScripPubKey locks a number of bitcoins into the output. Generally, a hash value is contained in the ScripPubKey. An address is obtained by hashing the hash value using SHA256 twice and then encoding it by Base58. For convenience, our discussion will focus on the previous transaction hash and previous output index of each input, the value of each output (quantity of bitcoins), and the bitcoin addresses obtained from ScriptPubKey of each output.

Bitcoin transactions fall into two categories: coinbase transactions and regular transactions. A coinbase transaction generates new bitcoins. It has no input, but has at least one output. A regular transaction transfers coins between addresses. It has at least one input and at least one output. As an example, Fig. 3 shows four transactions, where t_0 and t_1 are coinbase transactions, and t_2 and t_3 are regular transactions. The two inputs of t_3 correspond to *Output* of t_2 and *output[0]* of t_1. *Input[0]* of t_2 corresponds to *Output[0]* of t_0. For simplicity, Fig. 3 only shows the value and bitcoin address of each output. For example, *output[0]* of t_3 has a value of 1 and address a_0, which is extracted from the ScriptPubKey. For each input, Fig. 3 only shows the hash of the previous transaction and the output index of the previous transaction. For example, *input[0]* of t_3 includes *h2* (hash of transaction t_2) and 1 (the output index of t_2 corresponding to t_3's *input[0]*).

4.2. Construction of Bitcoin transaction nets

BTN, as the formal model of bitcoin transactions, is based on safe Petri nets. Given a list of bitcoin transactions, we use a transition (rectangle) to represent each bitcoin transaction, and a place (circle) to represent an input or output of each transaction. For each transaction, each input becomes an input place of the corresponding transition and each output becomes an output place of the corresponding transition. If an output of transaction t_i is used as input of another transaction t_j, the corresponding output place of transition t_i and the corresponding input of transition t_j are the same. Therefore, the given list of bitcoin transactions can be converted into one Petri net, where places capture the input/output relationships among the bitcoin transactions.

For each coinbase transition, we add an input place so that bitcoin transactions can be analyzed with standard Petri net semantics as a transition with no input place in Petri nets is always fireable. Fig. 4 shows the BTN that is corresponding to the transactions in Fig. 3. The transitions t_0, t_1, t_2 and t_3 represent the four transactions, respectively. Place p_0 represents *Output[0]* of transaction t_0 and *Input[0]* of transaction t_2. Place p_1 represents *Output[0]*

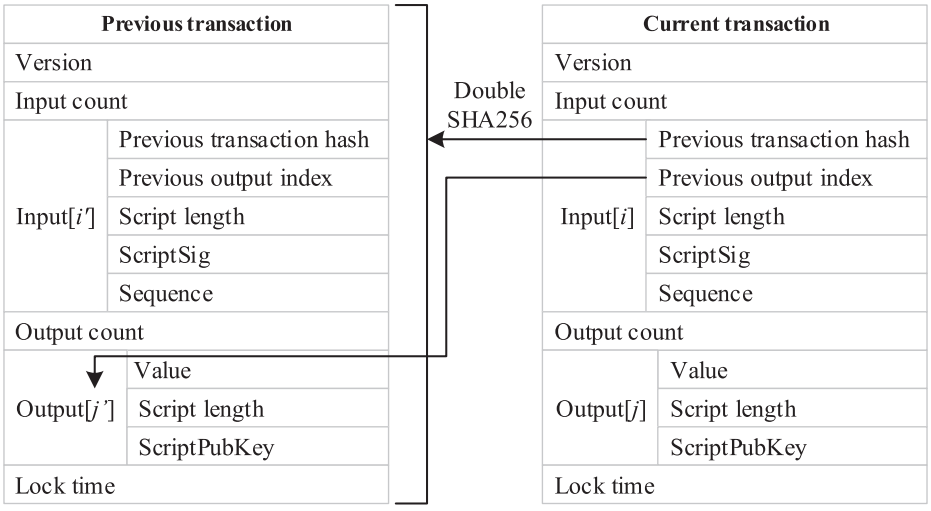


Fig. 2. Structure of bitcoin transactions.

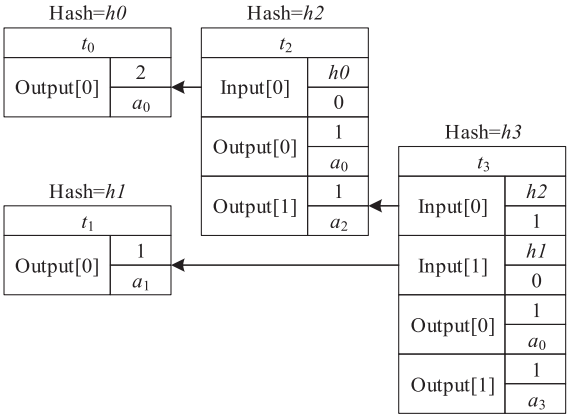


Fig. 3. Sample bitcoin transactions.

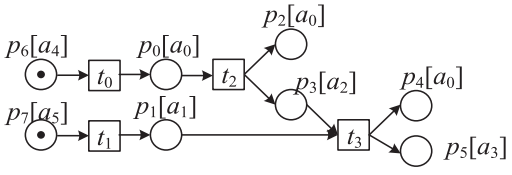


Fig. 4. Bitcoin transaction net for the transactions in Fig. 3.

of transaction t_1 and *Input* of transaction t_3. Places p_2 and p_3 represent *Output[0]* and *Output* of transaction t_2. respectively. p_3 is also corresponding to *Input[0]* of transaction t_3. Place p_4 and p_5 are *Output[0]* and *Output* of transaction t_3. Place p_6 (or p_7) is an added input for coinbase transaction t_0 (or t_1). Each place is annotated with a bitcoin address. If it represents a transaction's output, the address is extracted from the output's ScriptPubKey. If it is an added input of a coinbase transaction, the address is a unique random number.

We further extend safe Petri nets by associating transaction details with Petri net structures so that bitcoin transactions can be analyzed through transition firings. Formally, a BTN for a given list of bitcoin transactions is a 8-tuple $N=(P, T, F, A, V, \Psi, \tau, M_0)$, where P, T and A denote a finite set of places, a finite set of transactions, and a finite set of bitcoin addresses, respectively. $F\subseteq(P\times T)\cup(T\times P)$ is a set of arcs between places and transitions. V: $P\rightarrow R^{0+}$ is a value function on places, where R^{0+} is non-negative real number, and $V(p)$ is a bitcoin value. Ψ: $P\rightarrow A$ is an address mapping function on places and $\Psi(p)$ is an address associated with p. In a visualized Petri net, we use $p_i[a_j]$ to denote $\Psi(p_i)=a_j$. τ: $T\rightarrow time$ is a timestamp function on T and $\tau(t)$ is the timestamp of transition t. Since the blockchain data include the timestamps of blocks, rather than individual transactions, we use the timestamp of each block as the timestamp of all transactions contained in the block. M_0 is an initial marking, where $M_0(p)$ is the number (0 or 1) of tokens in place p – If p has no input transition (i.e., p is an input place of a coinbase transaction), then $M_0(p) = 1$, otherwise $M_0(p) = 0$. Transition t is said to be enabled or fireable under a marking if each input place p of t has at least one token. Firing an enabled transition removes the token from each input place and adds one token to each output place.

In BTN, A (bitcoin address space), V (bitcoin values in actual transactions), Ψ (addresses in actual transactions), τ (transaction timestamps) capture the detailed information of bitcoin transactions that is required of effective analysis of transaction flows. This information cannot be represented by the traditional Petri net structure (P, T, F, M_0), where tokens in markings (i.e., black dots) do not carry specific data. Although it is possible to extend high-level Petri nets to represent transaction information as structured data tokens, the complexity of analysis would be significantly increased. Thus, BTN in this paper is based on traditional safe Petri nets.

Note that multiple places (i.e., inputs/outputs of transactions) can be associated with the same address. When a change is returned to the same address of an input, the input and output places have the same address. In Fig. 3, t_2's *Output[0]* is a bitcoin change with an address of a_0. It is represented by p_2 in Fig. 4 and thus associated with a_0. As such, p_0 and p_2 have the same address a_0. The initial marking in Fig. 4 is [0, 0, 0, 0, 0, 0, 1, 1]: p_6 and p_7 have a token so that they can fire coinbase transitions t_0 and t_1, respectively. All other places have no token.

Algorithm 1 below describes how to transform a Bitcoin blockchain to a BTN.

Algorithm 1. Transform bitcoin transactions into a BTN.

Input: A block chain (a list of blocks).
Output: $(P, T, F, A, V, \Psi, \tau, M_0)$

1. For each block
2. For each bitcoin transaction t in the current block
3. Add t to T;
4. $\tau(t)$=block's timestamp;
5. For each output p of transaction t
6. Add p to P;
7. $M_0(p)$=0;
8. $V(p)$=*value* of output p;
9. Add (t, p) to F;
10. $\Psi(p)$=address a extracted from the ScriptPubKey field of output p;
11. Add a to A;
12. End for;
13. For each input p of transaction t
14. Add (p, t) to F;
15. End for;
16. End for;
17. End for
18. For each t in T
19. If t is coinbase transaction (i.e., $\bullet t=\varnothing$)
20. Create a new place p and add it to P;
21. $M_0(p)$=1;
22. Add (p, t) to F;
23. $V(p)=\sum_{p' \in {}^{\bullet}t} V(p')$;
24. Generate a new unique address a and add a to A;
25. $\Psi(p)=a$;
26. End if
27. End for

For analysis purposes, we introduce additional notations in Table 1. For example, In Fig. 4, $\bullet t_0 = \{p_6\}$, $t_0\bullet = \{p_0\}$, ${}^*t_0 = \{a_4\}$, $t_0{}^* = \{a_0\}$, $\bullet p_0 = \{t_0\}$, $p_0\bullet = \{t_2\}$, ${}^*a_0 = \{p_0, p_2, p_4\}$, $\bullet a_0 = \{t_0, t_2, t_3\}$ and $a_0\bullet = \{t_2\}$.

4.3. Bitcoin fluid

Bitcoins are transferred from addresses to addresses. We use fluid as an analogy of bitcoin flow. Colored bitcoin fluids are used to track and analyze bitcoin flows. There are three bitcoin fluid operations called dyeing, merging, and splitting. Initially, bitcoin fluids are colorless when bitcoins are locked in virtual addresses in BTN (in other words, before mined). An address is an analogy of a container of bitcoin fluids. When a transaction withdraws some bitcoins from the address, a splitting operation is needed to share bitcoin fluids from the container. When a transaction deposits some bitcoins into an address, a merging operation is needed to add and mix bitcoin fluids into the deposit address container. Fluids are mixed in the recipient container, but colors themselves do not mix. If we want to track and analyze bitcoins flow through a given address, this address is used as a dyeing container. The bitcoin fluid flowing into the address would be dyed by a specific color. The dyeing process is performed by the dyeing operation. When bitcoin fluids are transferred between addresses, the transferred bitcoin fluids are moved from one container to another.

Formally, a bitcoin fluid in a container C is defined as a triple $F=(c, q, w)$, where c is a color, q is a bitcoin quantity expressed in a non-negative real number, w is the percentage of q with respect to the total bitcoin quantity in this container. c is denoted as φ if c is colorless. A container C with several bitcoin fluids is defined as $\{(c_1, q_1, w_1), \ldots(c_n, q_n, w_n)\}$, where $w_i = q_i/\sum_j q_j$, and $\sum_j w_j = 100\%$.

The splitting operation is denoted as $Split(C, r)$, where r is bitcoin quantity needed to be split from container C. Its result is a container C' with some bitcoin fluids. Besides the returned result, the splitting operation also changes the input container C. Its algorithm is as follows.

Algorithm 2. Splitting operation

Input: C, r
Output: C'

1. $C'=\{\}$;
2. For each $((c_i, q_i, w_i)\in C)$
3. $\{C'=\{(c_i, r\times w_i, w_i)\}\cup C'$;
4. (c_i, q_i, w_i) in C is updated to $(c_i, q_i\text{-}r\times w_i, w_i)$;
5. If $(q_i\text{-}r\times w_i==0)$
6. $\{(d_i, q_i\text{-}r\times w_i, w_i)$ is removed from $C;\}\}$
7. Return C';

The merging operation, denoted as $Merge(C_1, C_2)$, merges C_1 and C_2 to C_3. It creates a new container C_3 that mixes together and holds all fluids in C_1 and C_2. Its algorithm is as follows.

Algorithm 3. Merging operation

Input: C_1, C_2
Output: C_3

1. $C_3=\{\}$;
2. For each (fluid $(c_i, q_i, w_i)\in C_1$)
3. {if $((c_i, q_j, w_j)\notin C_3)$
4. $\{C_3=\{(c_i, q_i, w_i)\}\cup C_3;\}$
5. Else
6. {update (c_i, q_j, w_j) to (c_i, q_i+q_j, w_j);}
7. For each (fluid $(c_i, q_i, w_i)\in C_2$)
8. {if $((c_i, q_j, w_j)\notin C_3)$
9. $\{C_3=\{(c_i, q_i, w_i)\}\cup C_3;\}$
10. Else
11. {update (c_i, q_j, w_j) to (c_i, q_i+q_j, w_j);}
12. Q=0;
13. For each (fluid $(c_i, q_i, w_i)\in C_3$)
14. $\{Q=Q+q_i;\}$
15. For each (fluid $(c_i, q_i, w_i)\in C_3$)
16. $\{w_i=q_i/Q;\}$
17. Return C_3;

The dyeing operation, denoted as $dye(C, c)$, dyes all fluids in container C by color c. Former colors of fluids in C are all replaced by the color c. Its algorithm is as follows.

Table 1
Notations of BTNs.

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

Algorithm 4. Dyeing operation

Input: C, c
Output: C

1. Q=0;
2. For each (fluid (c_i, q_i, w_i)$\in C$)
3. {Q=Q+q_i;}
4. C={(c, Q, 100%)};
5. Return C;

An address a has two features about bitcoin quantities, balance and total received coin amount, which are denoted as *a.balance* and *a.received*. Therefore, the address a has two containers of bitcoin fluids, one is for the *a.balance*, another is for the *a.received*. the two containers are denoted as *a.balance.container* and *a.received.container*. Algorithm 5 below describes how to generate container features of addresses.

Algorithm 5. Analysis of Coin Distribution

Input: a BTN=(P, T, F, A, Ψ, τ, M_0), a set of interesting address A'={a_{i1}, a_{i2}……} with their color names.
M: current marking; T': fireable transition set; C: temp container;
Output: for $\forall a\in A'$, *a.balance.container*, *a.received.container*

1. M=M_0;
2. For $\forall a \in A$
3. *a.balance.container* =∅;
4. *a.received.container* =∅;
5. End for
6. For $\forall a\in\{a|a=\Psi(p)\wedge|\bullet p|=0\}$
7. *a.balance.container* ={(ϕ, *a.balance*, 100%)};
8. *a.received.container* ={(ϕ, *a.balance*, 100%)};
9. End for
10. T'={$t|M(p)$=1$\wedge\forall p\in\bullet t$};
11. While $T'\neq\emptyset$ do
12. Find $t\in T'$ such that $\tau(t)\leq\tau(t_i)$ for $\forall t_i\in T'\wedge t\neq t_i$;
13. C=∅;
14. For $\forall p\in\bullet t$
15. $M'(p)$=$M(p)$-1;
16. C'=*Split*($\Psi(p)$.*balance.container*, $V(p)$);
17. *C=Merge(C,C');*
18. End for
19. For $\forall p\in t\bullet$
20. $M'(p)$=$M(p)$+1;
21. C'=*Split*(C, $V(p)$);
22. If $\Psi(p)\in A'$
23. C'=*Dye*(C', $\Psi(p)$.*dyeingcolor)*;
24. Endif
25. $\Psi(p)$.*balance.container* =*Merge*($\Psi(p)$.*balance.container, C'*);
26. $\Psi(p)$.*received.container* =*Merge*($\Psi(p)$.*received.container*, *C'*);
27. End for
28. M=M';
29. T'={$t|M(p)$=1$\wedge\forall p\in\bullet t$};
30. End while

When a transition fires (line 12), for each input p (line 14), fluid C' of $V(p)$ is split from balance container of address$\Psi(p)$ (line 16). After that C' is merged into container C (line 17). For each output p (line 19), container C' of $V(p)$ is split from C (line 21). If $\Psi(p)$ is a dyeing address (line 22), C' is dyed by the dyeing color of $\Psi(p)$ (line 23). Then the balance and total received containers of $\Psi(p)$ are merged with C' respectively (lines 25 and 26).

5. Analysis of Bitcoin transactions

This section describes extraction of information about the features of bitcoin transactions, expression and analysis of transaction patterns, and visualization techniques for analysis of suspected bitcoin addresses.

5.1. Extraction of feature information

Once the BTN is constructed, we can obtain information about various features for forensic analysis, as shown in Table 2. The first 14 features are expressed with respect to the BTN structure. The remaining features are related to flows of bitcoin transactions. They are determined by traversing all transactions according to the firing of the BTN.

Transaction time refers to the time when the transaction occurred. Coin amount of a place p is parsed from the *value* field in the related *output[i]* of a transaction. Input (output) place count of a transaction represents the number of input (output) places in the transaction. Input/output address count of a transaction represents the number of addresses that are mapped by input/output places of the transaction. A transaction used to gather coins usually has numerous input places/addresses. A transaction used to split coins usually has numerous output addresses/places. Transferred coin amount refers to the number of coins that were transferred by the transaction. Place count of an address denotes the number of inputs/outputs (places) that are mapped to the address. Deposit/withdraw transaction counts of an address denotes the number of transactions that deposit or withdraw coins to or from the given address. First transaction time and last transaction time of an address are the first and last transactions' time stamps related with the address. Incoming address count of an address denotes the number of addresses that transfer coins to the given address. Outgoing address count denotes the number of addresses that received transferred coins from the given address. A gathering address usually has a greater number of incoming addresses than outgoing addresses. Besides above features, *a.balance*, *a.balance.container*, *a.received* and *a.received.container* have been discussed in the last section. All of these features can be used to define specific transaction patterns. Next, we will illustrate how to obtain features

Table 2
List of features.

No	Feature	Formal expression
1	Transaction time of transaction t	$\tau(t)$
2	Coin amount of transaction output p	$V(p)$
3	Number of inputs of transaction t	$\lvert\bullet t\rvert$
4	Number of outputs of transaction t	$\lvert t\bullet\rvert$
5	Number of input addresses of transaction t	$\lvert{}^*t\rvert$
6	Number of output addresses transaction t	$\lvert t^*\rvert$
7	Transferred coin amount of transaction t	$\sum_{p\in t\bullet}V(p)$
8	Number of times that address a has occurred in the transaction outputs	$\lvert{}^*a\rvert$
9	Number of deposit transactions of address a	$\lvert\{t\lvert t\in\bullet a\}\rvert$
10	Number of withdraw transactions of address a	$\lvert\{t\lvert t\in a\bullet\}\rvert$
11	First transaction time of address a	$\tau_f(a)=\tau(t)$ such that $\tau(t)\leq\tau(t_i)$ for t, $t_i\in\bullet a\cup a\bullet$
12	Last transaction time of address a	$\tau_L(a)=\tau(t)$ such that $\tau(t)\geq\tau(t_i)$ for t, $t_i\in\bullet a\cup a\bullet$
13	Number of incoming addresses that transfer coins to address a	$\lvert\{a'\lvert p\in{}^*a\wedge t\in\bullet p\wedge a'\in{}^*t\}\rvert$
14	Number of outgoing addresses to which coins are transferred from address a	$\lvert\{a'\lvert p\in{}^*a\wedge t\in p\bullet\wedge a'\in t^*\}\rvert$
15	Coin balance of an address a	*a.balance*
16	Total received coin amount of an address a	*a.received*
17	Distribution of balance coins	*a.balance.container*
18	Distribution of received coins	*a.received.container*

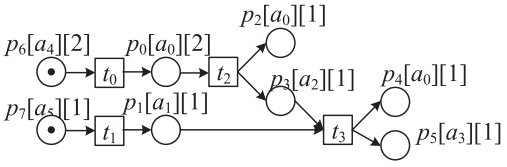


Fig 5. The BTN in Fig. 4. Its places are labeled with $V(p)$.

related bitcoin flow.

Consider Fig. 5 as an example. Notation $p[\Psi(p)][V(p)]$ is labeled by side of place p. $\Psi(p)$ is an address mapped by the place p, and $V(p)$ is bitcoin quantity moved by p. Suppose a_0 is the dyeing address and its dyeing color is c_0. We want to know which addresses hold the bitcoins that passed through a_0 and the percentage amounts in holding address balances. Assume the firing sequence is $t_0t_1t_2t_3$ and the corresponding sequence of states is $M_0M_1M_2M_3M_4$. In M_0, for each coinbase transition t, the balance/received container of address $\Psi(p)$ as t's input place $p\in\bullet t$ is initialized to $\{(\varphi, V(p), 100\%)\}$ and the balance/received container of address $\Psi(p)$ as t's output place $p\in t\bullet$ is initialized to an empty set. The balance/received container of any address at an input or output place of a non-coinbase transition is initialized to an empty set. Therefore, the balance containers of a_4 and a_5 are $\{(\varphi, 2, 100\%)\}$ and $\{(\varphi, 1, 100\%)\}$ respectively.

When t_0 fires, $C'=(\varphi, 2, 100\%)$ is split from the balance container of a_4 and the balance container of a_4 becomes $\varnothing$. Since a_0 is a dyeing address, C' is dyed to $(c_0, 2, 100\%)$ and merged into the balance/received container of a_0. Then the balance/received container of a_0 becomes $(c_0, 2, 100\%)/(c_0, 2, 100\%)$.

When t_1 fires, $C' = \{(\varphi, 1, 100\%)\}$ is split from a_5's balance container and a_5's balance container becomes $\varnothing$. Then C' is merged into the balance/received container of a_1. Then the balance/received container of a_1 becomes $\{(\varphi, 1, 100\%)\}/\{(\varphi, 1, 100\%)\}$.

When t_2 fires, $C_0' = \{(c_0, 2, 100\%)\}$ is split from the balance container of a_0 and the balance container of a_0 becomes $\varnothing$. For output place p_2, $C_0' = \{(c_0, 1, 100\%)\}$ is split from C' since a_0 is a dyeing address, C_0' is dyed to $\{(c_0, 1, 100\%)\}$ and merged into the balance/received container of a_0. Therefore, the balance/received container of a_0 is $\{(c_0, 1, 100\%)\}/\{(c_0, 3, 100\%)\}$. For output place p_3, $C_2' = \{(c_0, 1, 100\%)\}$ is split from C' and merged into the balance/received container of a_2. Therefore, the balance/received container of a_2 is $\{(c_0, 1, 100\%)\}/\{(c_0, 1, 100\%)\}$.

When t_3 fires, $C_1' = \{(\varphi, 1, 100\%)\}$ and $C_2' = \{(c_0, 1, 100\%)\}$ are split from balances of C_1 and C_2 respectively. The balances of C_1 and C_2 become $\varnothing$ and $\varnothing$. C_1' and C_2' are merged into C and $C = \{(\varphi, 1, 50\%), (c_0, 1, 50\%)\}$. For output place p_4, $C_4' = \{(\varphi, 0.5, 50\%), (c_0, 0.5, 50\%)\}$ is split from C. Since a_0 is a dyeing address, C_4' is dyed to $\{(c_0, 1, 100\%)\}$ and merged into the balance/received container of a_0. Therefore, the balance/received container of a_0 is $\{(c_0, 2, 100\%)\}/\{(c_0, 4, 100\%)\}$. For output place p_5, $C_5' = \{(\varphi, 0.5, 50\%), (c_0, 0.5, 50\%)\}$ is split from C and merged into the balance/received container of a_3. Therefore, the balance/received container of a_3 is $\{(\varphi, 0.5, 50\%), (c_0, 0.5, 50\%)\}/\{(\varphi, 0.5, 50\%), (c_0, 0.5, 50\%)\}$.

The analysis of balance fluids indicates that the bitcoins that flow out a_0 are held by a_0 and a_3, in state M_4. In a_3, the bitcoins from a_0 account for 50% of its balance. The analysis of total received coins indicates that the bitcoins passed through a_0 once were received by a_0, a_2 and a_3. The bitcoins from a_0 account for 100% and 50% of the total received coins in a_2 and a_3 respectively.

5.2. Pattern matching

A pattern is a defined by a set of properties over features according to the clues provided. Pattern matching aims to find addresses that satisfy the given properties. A property is a logical expression of features. Different feature expressions can be combined to form a complex pattern expression. A pattern expression E can be used to describe a set of addresses $\{a|a$ satisfies $E\}$. For features 1–16 in Table 2, a basic pattern expression specifies a range of a feature. For example, $0 < a.balance<100$ results in a set of addresses with a balance greater than 0 and less than 100, i.e., $\{a|0 < a.balance<100\}$. Different features can be combined to a complex expression. For example, the expression $(a.balance<10)\wedge(\sum_{p\in{}^*a}V(p)>100)$ results in a set of addresses with a balance of less than 10 and total received coins greater than 100. Features 17 and 18 are fluids features of addresses. Fluids features require two conjunction clauses to relate them to addresses, which are $F\in a.balance.container$ and $F\in a.received.container$. For example, if we want to find addresses with balances that contain a given color c', the expression is $(c' = F.c)\wedge(F\in a.balance.container)$ which results $\{a|(c' = F.c)\wedge(F \in a.balance.container)\}$.

Transaction features 1–7 are used to match transactions. To find suspicious addresses, we use conjunction clauses ($t\in\bullet a$ and $t\in a\bullet$) to relate transactions to addresses. Suppose we need to find the addresses with transaction time before 01/01/2010. The expression $(\tau(t) < 01/01/2010)\wedge(t\in\bullet a\vee t\in a\bullet)$ results in an address set

Table 3
Elements of visualizations.

Feature	Visualization Representation	Visualization Element
Total received coin amount	Node Size	Node size indicates total received coin amount relative to other nodes visualized. Larger node size indicates greater total received coin amount and smaller node size indicates less total coin received amount.
Incoming address count	Nodes surrounding a target node	The number of nodes with **blue** directed edges from self to target node is the incoming address count
Outgoing address count	Nodes surrounding a target node	The number of nodes with **red** directed edges from target node to self is the outgoing address count
Transaction Transferred coin amount	Edge Color and Edge Label	Transaction transferred coin amount is an edge color gradient relative to other edges visualized One-Layer Input Edge Less Transferred Coin More Transferred Coin One-Layer Output Edge Less Transferred Coin More Transferred Coin Two Layer Input/Output Edge Less Transferred Coin More Transferred Coin

$\{a|(\tau(t) < 01/01/2010) \wedge (t \in \bullet a \vee t \in a\bullet)\}$. If we want to find the addresses that their deposit transaction time early than 01/01/2010, we can use the expression $(\tau(t) < 01/01/2010) \wedge (t \in \bullet a)$. $p \in \bullet t$ and $p \in t\bullet$ can be used to relate addresses to transactions. For example, if we want to find addresses that have withdrawal transactions with an output containing more than 100 bitcoins, the expression is $(V(p) > 100) \wedge (p \in t\bullet) \wedge (t \in a\bullet)$, which results in $\{a|(V(p) > 100) \wedge (p \in t\bullet) \wedge (t \in a\bullet)\}$.

5.3. Visualization

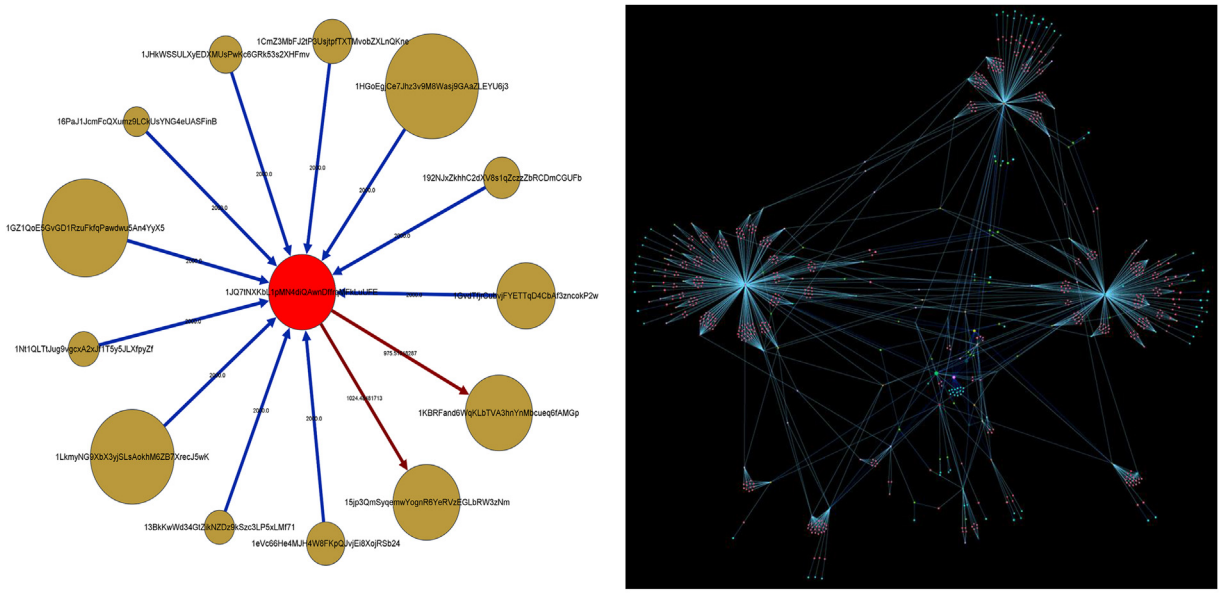
We created a stand-alone, automated visualization tool using the open source Gephi Toolkit (Gephi.org, 2017) and Blockchain.Info API (Bitcoin, 2017). This tool creates visualizations of the flow of Bitcoins at varying user-specified transaction depths (layers) through a target address. This was used to visualize the flow of bitcoins through the suspected gathering addresses as a method of finding incorrect or neglected patterns. Table 3 shows the elements of flow visualization that represent several of the features that we defined for transaction pattern analysis.

In the visualization, each node represents a bitcoin address and the directed edges represent the directional flow of Bitcoins as part of transactions from one address (node) to another address (node). Note that the target address in the visualization is always red and that several disjoint nodes and edges may be part of one transaction with multiple inputs or outputs. Fig. 6 shows two examples of one-layer and two-layer visualizations.

6. A case study

6.1. The Mt.Gox case

Mt.Gox was a bitcoin exchange launched in July 2010. In February 2014, Mt.Gox filed for bankruptcy protection due to the



One Layer Visualization Two Layer Visualization

Fig. 6. Visualizations of the address 1JQ7tNXKbL1pMN4diQAwnDffrqMFkLuUFE.

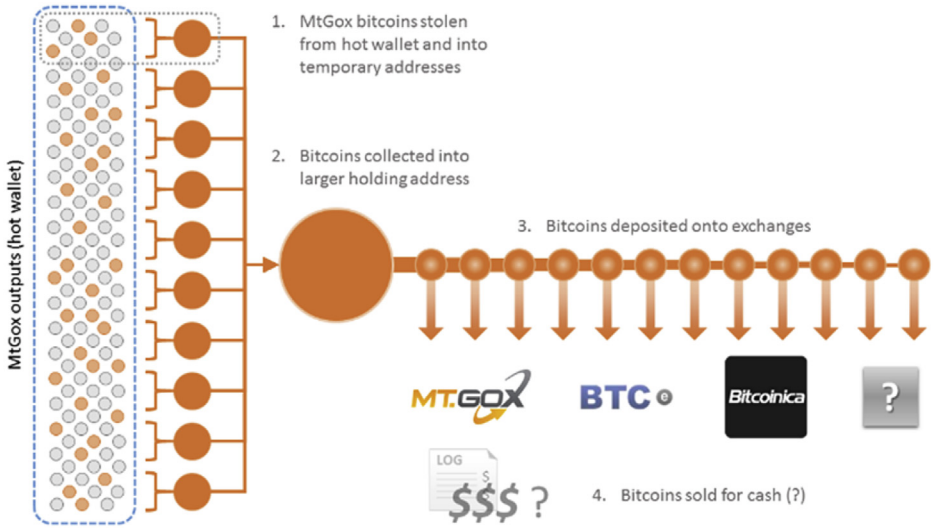


Fig. 7. A recurring transfer pattern (WizSec, 2015) of the lost bitcoins form Mt.Gox.

loss of more than 500 k bitcoins. Bitcoin loss most likely began in Aug. 2011 and lasted until late 2013. Based on the analysis of a Mt.Gox deposit and withdraw log data leaked in 2014 (which is no longer publicly available now), WizSec identified a transfer pattern as shown in Fig. 7. The goal of our case study is to identify gathering addresses that match the transfer pattern. These addresses were likely used to transfer the lost bitcoins.

6.2. Experiment setup

The timeframe of bitcoin transactions under investigation is January 3, 2009–December 28, 2013 and this contains more than 30,000,000 transactions. We used the open source tool BitcoinDatabaseGenerator (ladimolnar, 2017) to parse the transactions and save them into a database. This was conducted on a PC with an eight core server CPU and 64 GB RAM. It took approximately 20 min to transform the transactions into a BTN and approximately 40 min to complete the analysis.

6.3. Pattern analysis

According to WizSec's report, "MtGox bitcoins would suddenly get sent to a new non-MtGox address, without any withdrawal log entry, often in fairly recognizable amounts of a few hundred BTC at a time. Shortly afterwards, these addresses in turn would get 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." This is reflected by the transfer pattern in Fig. 8. Our analysis aims to find these large holding addresses, which are called gathering addresses.

According to the clues indicated by WizSec, we formalized the transfer pattern with the following five expressions, where a represents a gathering address.

R1. The number of incoming addresses of a gathering address is greater than or equal to 3. Formally, $|\{a'|p\in{}^{*}a\wedge t\in\bullet p\wedge a'\in{}^{*}t\}|\geq 3$.
R2. Each gathering address holds a few thousand bitcoins. This is expressed as $1000 \leq a.received \leq 10{,}000$.
R3. Bitcoin loss behavior started in August 2011. For the sake of safety, we set the start date to July 15, 2011. This clue is expressed as $(t\in\bullet a\vee t\in a\bullet)\wedge(\tau(t)\geq \text{July/15/2011})$.
R4. Every transaction that transferred bitcoins to a large holding address a has only one output place. The expression is $(t\in\bullet a)\wedge(|t\bullet|=1)$.
R5. The number of outgoing addresses of a gathering address a is at least 2, i.e.,$|\{a'|p\in{}^{*}a\wedge t\in p\bullet\wedge a'\in t^{*}\}|\geq 2$. There are two methods to accept coin change: using the gathering address, and using a new change address. It is not known which method was used in the Mt.Gox case. If the first was used, there would be many outgoing addresses. 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 is 2.

The above analysis resulted in 265 gathering addresses which transferred 592 K bitcoins. We visualized the connections of each of the gathering addresses with other addresses at one level transaction depth using our visualization tool. Fig. 8 shows a pattern of suspicious gathering addresses, where the center node is a suspicious gathering address, which has many input addresses (nodes

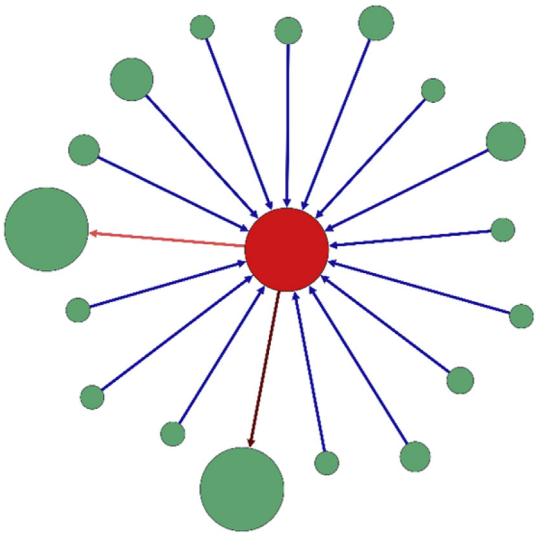


Fig. 8. A pattern of gathering addresses.

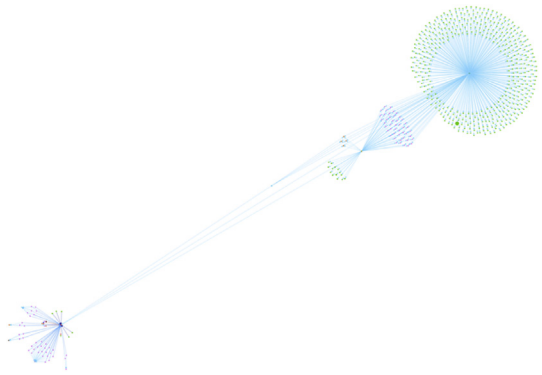


Fig. 9. A pattern found among addresses incorrectly marked as suspicious gathering addresses.

connected with blue edges) and a small number (mostly two) of output addresses (nodes connected with red nodes). This is consistent with the clues in Fig. 7.

Several addresses were found to have the pattern shown in Fig. 9. This was unexplainable with the recurring transfer pattern. We checked these gathering addresses and found that the limitation of the number of incoming addresses of withdraw transaction was neglected. In the pattern in Fig. 7, the withdraw transaction withdrew bitcoins from only one gathering address, while the withdraw transaction in Fig. 9 withdrew bitcoins from two addresses, which is inconsistent with the pattern. Therefore, a sixth rule is added as follows.

R6. Every transaction that withdraws bitcoins from a gathering address has only one incoming address. This is represented by $(t \in a\bullet) \wedge ({}^{*}t = 1)$.

After the application of this expression, the number of gathering addresses was reduced to 227. All of them follow the pattern in Fig. 8. These addresses transferred a total of 527 K BTC, which was close to 500 K. Note that not all addresses that match the pattern in Fig. 7 were necessarily involved in the Mt.Gox case. To identify which addresses are involved with Mt.Gox, the colored fluid method was used to analyze the bitcoin flow. '1LNWw6yCxkUmkhArb2Nf2MPw6vG7u5 WG7q' is a well-known Mt.Gox address. It is used as a dyeing address. The Mt.Gox fluid spreads from the dyeing address. A suspected address containing the Mt.Gox fluid indicates that it received some bitcoins from Mt. Gox. Thus we introduced a seventh rule:

R7. The container of total received coins of a gathering addresses contains Mt.Gox fluid. This is represented by $F \in a.received.container \wedge F.c = Mt.Gox$.

This reduced the number of gathering addresses to 187, which transferred a total of 464 K bitcoins. Note that, although WizSec has identified many possible Mt.Gox addresses, these addresses are not publicly available. In this paper, we have used only a single known Mt.Gox address as the dyeing address and this is likely the reason why our identified bitcoin amount is less than 500 K.

Fig. 10 presents the distribution of Mt.Gox fluid weight in the total received coins of the suspected gathering addresses. The number of gathering addresses with less than 10% Mt.Gox fluid is 19; the number of gathering addresses with 10%–20% Mt.Gox fluid is 37; the number of gathering addresses with 20%–30% Mt.Gox fluid is 102; the number of gathering addresses with more than 30% Mt.Gox fluid is 29. Most gathering addresses contain more than 10% Mt.Gox fluid where this 10% value indicates that more than 10% of its received bitcoins come from the known Mt.Gox address. The higher the percentage of Mt.Gox fluid in a given address, the more suspicious it is. The experimental results show that most addresses contain a certain percentage of Mt.Gox fluid.

6.4. Further analysis of suspected gathering addresses

Visualization techniques were employed to further investigate shared characteristics and connections among the 187 suspicious gathering addresses. Using the one-layer visualization, we grouped the addresses by the number of incoming addresses (number of input addresses/nodes) and the number of outgoing addresses. This revealed a subgroup of 16 addresses (Fig. 11) with the following mutual characteristics.

- Each address has a total received coin amount of 3000 ± 0.001 BTC.

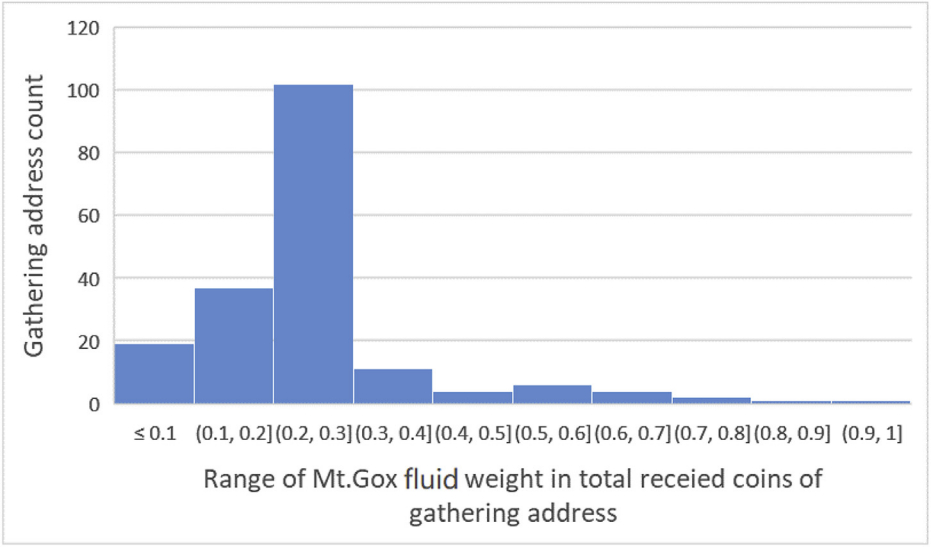


Fig. 10. Ditribution of Mt.Gox fluid weight on gathering addresses.

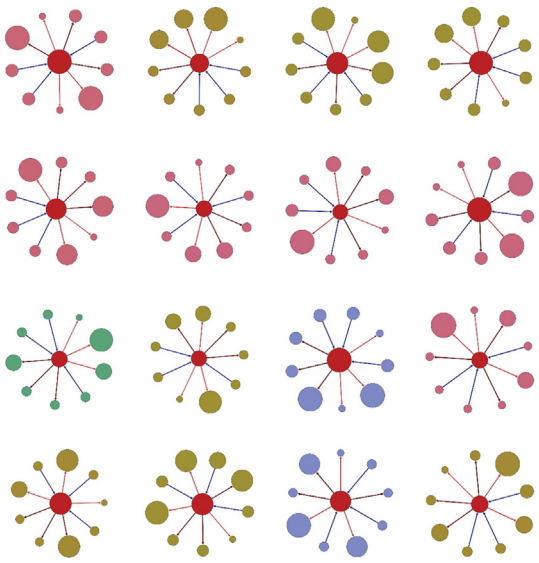


Fig. 11. One-layer visualizations of the 16 addresses.

- Each address has exactly 3 input transactions and 6 output transactions: receive 1000 BTC from one address (t_1), send 1000 BTC to two addresses (t_2), receive 1000 BTC from one address (t_3), send 1000 BTC to two addresses (t_4), receive 1000 ± 0.001 BTC from one address (t_5), send 1000 ± 0.001 BTC to two addresses (t_6).
- t_1 and t_2 (t_3/t_4, t_5/t_6) place on the same day.
- The first transaction time of all addresses occurred between 9/21/2013 and 11/30/2013.
- The last transaction time of all addresses occurred between 12/19/2013 and 12/22/2013.
- The distribution of Mt.Gox fluid weight has a median value of 0.1845 with a max of 0.2336 and min of 0.1298.

Due to the numerous similarities between the 16 addresses within this subgroup, we believe that they may be controlled by a single user or group for a single purpose. We also applied the two-layer transaction depth visualization of all 187 suspected gathering addresses as shown in Fig. 12. Nodes of suspected gathering addresses are shown in red and the node of the Mt.Gox dyeing address used in our analysis is shown in green. From this visualization, we found that 180 of the 187 suspected gathering addresses were connected. However, no mutual characteristics were identified within this subgroup of 180 addresses or within the subgroup of the remaining 7 addresses. It seems that the majority of the suspected gathering addresses are linked to each other through outputs or inputs at various levels of connection.

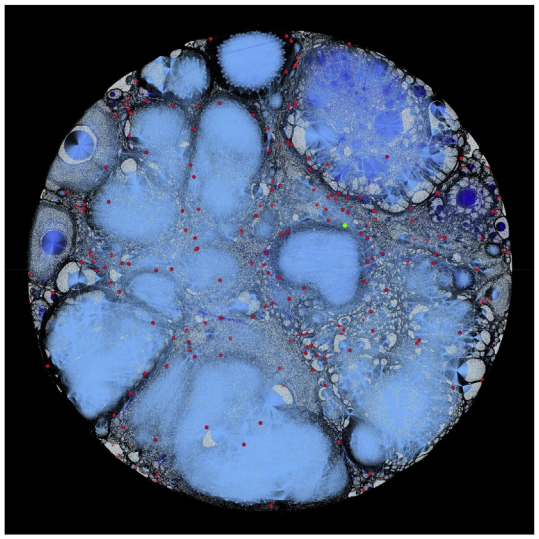


Fig. 12. All 187 suspected addresses at a depth of two visualized together.

7. Conclusions

We have presented the FABT framework for forensic analysis of bitcoin thefts by identifying suspicious addresses with transaction patterns. The main contributions of this paper are: (a) BTN as a modeling formalism for bitcoin transactions, which allows for specification and analysis of transaction patterns based on 19 static and dynamic features, (b) the Bitcoin fluid analysis method for tracking bitcoin flows, and (c) the application of the proposed approach to the real-world Mt.Gox case.

In the Mt.Gox case study, we have used the only one publicly known address as the dyeing address to track transaction flows. The resultant suspected addresses do not necessarily cover all addresses involved in transfer of the stolen bitcoins. One possible improvement is that we can use clustering algorithms to find additional addresses that have other similar behaviors to these suspected addresses. Another possibility is to find additional addresses by using the suspected addresses as dyeing addresses in the bitcoin fluid method. Our future work will also investigate various patterns of abnormal bitcoin transactions, such as malicious transactions for the purposes of price manipulation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.fsidi.2019.200895.

References

Androulaki, E., Karame, G.O., Roeschlin, M., Scherer, T., Capkun, S., 2013. Evaluating user privacy in bitcoin. In: Proceedings of *International Conference on Financial Cryptography and Data Security*, Berlin, Heidelberg, pp. 34–51.

Battista, G.D., Donato, V.D., Patrignani, M., Pizzonia, M., Roselli, V., Tamassia, R., 2015. Bitconeview: visualization of flows in the bitcoin transaction graph. In: Proceedings of *IEEE Symposium on Visualization for Cyber Security*. VizSec, pp. 1–8.

Bistarelli, S., Santini, F., 2017. Go with the -bitcoin- flow, with visual analytics. In: The Proceedings of the 12th International Conference on Availability, Reliability and Security. Reggio Calabria, Italy.

Bitcoin, B.L.S.A., 2017. Developer APIs. Available: https://www.blockchain.com/api.

Christin, N., 2013. Traveling the silk road: a measurement analysis of a large anonymous online marketplace. In: The Proceedings of the 22nd international conference on World Wide Web, Rio de Janeiro, Brazil.

Di Francesco Maesa, D., Marino, A., Ricci, L., 2018. Data-driven analysis of Bitcoin properties: exploiting the users graph. Int. J. Data Sci. Anal. 6 (1), 63–80.

Feder, A., Gandal, N., Hamrick, J.T., Moore, T., 2017. The impact of DDoS and other security shocks on Bitcoin currency exchanges: evidence from Mt. Gox. J. Cybersecur. 3, 137–144.

Fleder, M., Kester, M.S., Pillai, S., 2015. Bitcoin transaction graph analysis. 1502.01657.

Gephiorg, 2017. Gephi Toolkit. Available: https://gephi.org/toolkit/.

Göbel, S., 2016. A Polynomial Translation of Mobile Ambients into Safe Petri Nets: Understanding a Calculus of Hierarchical Protection Domains. Springer.

Harlev, M.A., Sun Yin, H., Langenheldt, K.C., Mukkamala, R., Vatrapu, R., 2018. 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, pp. 1–10.
Kondor, D., Pósfai, M., Csabai, I., Vattay, G., 2014. Do the rich get richer? An empirical analysis of the Bitcoin transaction network. PLoS One 9, 1–10.
ladimolnar, 2017. BitcoinDatabaseGenerator. Available: https://github.com/ladimolnar/BitcoinDatabaseGenerator.
McGinn, D., Birch, D., Akroyd, D., Molina-Solana, M., Guo, Y., Knottenbelt, W.J., 2016. Visualizing dynamic bitcoin transaction patterns. Big Data 4, 109–119.
Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G.M., et al., 2013. A fistful of bitcoins: characterizing payments among men with no names. In: The Proceedings of the 2013 Conference on Internet Measurement Conference. Barcelona, Spain.
Monaco, J.V., 2015. Identifying Bitcoin users by transaction behavior. In: Proceedings of *SPIE Defense + Security*, p. 15.
Möser, M., Böhme, R., Breuker, D., 2013. An inquiry into money laundering tools in the Bitcoin ecosystem. In: Proceedings of *APWG eCrime Researchers Summit*, pp. 1–14.
Nakamoto, S., 2008. Bitcoin: A Peer-To-Peer Electronic Cash System.
Pinna, A., Tonelli, R., Orrú, M., Marchesi, M., 2017. A Petri nets model for blockchain analysis. 1709.07790.
Reid, F., Harrigan, M., 2011. 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, pp. 1318–1326.
Ron, D., Shamir, A., 2013. Quantitative analysis of the full bitcoin transaction graph. In: Proceedings of *International Conference on Financial Cryptography and Data Security* Berlin. Heidelberg, pp. 6–24.
Ron, D., Shamir, A., 2014. How did Dread pirate Roberts acquire and protect his bitcoin wealth?. In: Proceedings of *International Conference on Financial Cryptography and Data Security*, Berlin, Heidelberg, pp. 3–15.
Spagnuolo, M., Maggi, F., Zanero, S., 2014. Bitlodine: extracting intelligence from the bitcoin network. In: Proceedings of *International Conference on Financial Cryptography and Data Security*, Berlin, Heidelberg, pp. 457–468.
WizSec, 2015. The missing MtGox bitcoins. Available: https://blog.wizsec.jp/2015/04/the-missing-mtgox-bitcoins.html.