2018 IEEE International Conference on Data Mining Workshops (ICDMW)

BiVA: Bitcoin Network Visualization & Analysis

Frédérique Oggier
Division of Mathematical Sciences
Nanyang Technological University, Singapore
Email: frederique@ntu.edu.sg

Silivanxay Phetsouvanh and Anwitaman Datta
School of Computer Science and Engineering
Nanyang Technological University, Singapore
Email: silivanx002@e.ntu.edu.sg, anwitaman@ntu.edu.sg

Abstract—**We showcase a graph mining tool, BiVA, for visualization and analysis of the Bitcoin network. It enables data exploration, visualization of subgraphs around nodes of interest, and integrates both standard and new algorithms, including a general algorithm for flow based clustering for directed graphs, and other Bitcoin network specific wallet address aggregation mechanisms. The BiVA user interface makes it easy to get started with a basic visualization that gives insights into nodes of interests, and the tool is modular, allowing easy integration of new algorithms. Its functionalities are demonstrated with a case study of extortion of Ashley Madison data breach victims.**

I. Introduction

Cryptocurrencies, and in particular Bitcoin, have become immensely popular in recent years. They rely on a public ledger that allows anyone to check the chain of transactions, yet they provide anonymity of their users due to their decentralized nature. The pseudo-anonymous nature of cryptocurrencies has made them the de facto medium for online black markets, money laundering and extortionists (e.g., Ashley Madison data breach related blackmail [1], WannaCry ransomware extortion [2]). At the same time, the public ledger provides information of interest for financial analysts, as well as for police and forensics analysts tracking illegal activities.

Early tools for Bitcoin analysis are plenty, e.g. [3], [4], [5], [6], [7], [8], exploiting simple heuristics clubbing co-paying addresses, and guessing the address used to collect the change in a transaction (e.g., [9], [10]). The introduction of mixing, where multiple users come together (possibly via a service, or in a decentralized manner) to create multi-input multi-output transactions render the simple heuristics ineffective, by making it difficult to trace the flow of money, since mixing obfuscates the linkage among inputs and outputs of a transaction. It is in fact demonstrated in [6] that clubbing multiple inputs together in a cluster is surprisingly effective, but false positives are indeed caused by transactions which involve mixing. Deanonymization techniques have been tried (e.g., [11], [12], [13]), enhanced by side ('off-chain') information [11], [14] obtained from web crawling and known addresses of businesses involved in the Bitcoin ecosystem (e.g., mining pools, exchanges) to possibly identify real world identities.

Tools for exploratory data analysis can help shed light on patterns, to complement forensics for deanonymization of Bitcoin users, as well as to understand social or financial behaviors [15]. This motivated the design of BiVA, a graph mining tool to aide visualization and analysis of information from the Bitcoin network.

There are numerous works (e.g. [5], [8], [17]) that analyze and generate infographics, often of macroscopic aspects of cryptocurrencies, such as the variation of price and volume of transactions over time. Instead, BiVA starts from a given Bitcoin transaction or wallet address of interest, around which a portion of the Bitcoin network is zoomed into, taking into account a desired perimeter, time window, or Bitcoin amounts. The subgraph of interest is then analyzed, through classical graph algorithms as well as new graph algorithms (described in Subsection III-B) motivated in particular by the proliferation of mixing transactions in the recent years. The latter include information flow based centrality and clustering, capturing the likelihood of Bitcoin flow confined within clusters; a path confluence based address aggregation technique; and a mechanism for estimating the flow of money by traceback/forward to/from a set of nodes of interests.

The tools [4], [18] are the closest works to BiVA: [4] supports the simple heuristic of address aggregation, as well as tracing how the money flows among such aggregated address clusters. [18] provides some simple queries over the Bitcoin blockchain dataset, for instance, it identifies miners (e.g. from transactions that have no inputs), looks at all the transactions a specific address is involved in, looks at all addresses involved in a specific transaction, or it finds connected components on the transaction graph. But [18] involves no particular analysis, and can in fact be directly queried from the blockchain. This aspect is in essence similar to what numerous web based blockchain navigation services such as [19], [20] provide.

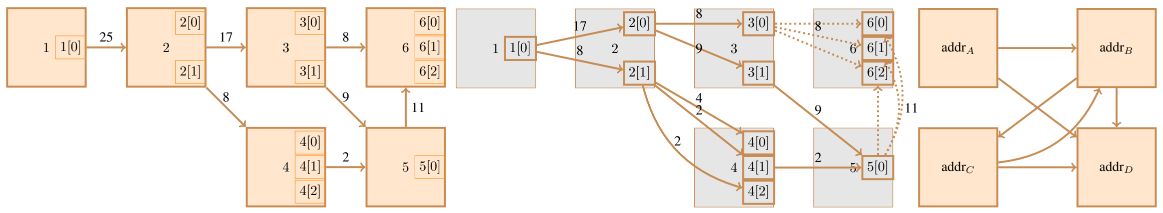
Finally, BiVA's interface and modularity are meant to be easy to use, irrespectively of whether one just wants to visually explore particular Bitcoin transactions, or more seriously do forensics and possibly add extra features. We illustrate its application, in Section IV, to the Ashley Madison Bitcoin scam, which happened in the second half of the year 2015, after the introduction of mixing services. Out of the approx. 45 million Bitcoin transactions during that year, approx. 3.4 millions of them could involve mixing, a significant volume,, justifying the need for algorithms developed to enable BiVA.

We next provide some background on the Bitcoin network, to position our data model and system architecture design.

II. Background: The Bitcoin Network

The Bitcoin network is a peer-to-peer payment network, where users send and receive a cryptocurrency called Bitcoin, by broadcasting digitally signed messages to the network.

2375-9259/18/$31.00 ©2018 IEEE
DOI 10.1109/ICDMW.2018.00210



(a) The transaction only network. (b) The (transaction) input-output network. (c) The (wallet) address network.

Fig. 1: Different network views to capture Bitcoin transactions. Dotted lines indicate ambiguity, the amount indicated then refers to the sum of the dotted lines.

1	Inputs: ∅ Outputs: 25.0 → addr$_A$
2	Inputs: 1[0] Outputs: 17.0 → addr$_B$, 8.0 → addr$_A$
3	Inputs: 2[0] Outputs: 8.0 → addr$_C$, 9.0 → addr$_B$
4	Inputs: 2[1] Outputs: 4.0 → addr$_D$, 2.0 → addr$_B$, 2.0 → addr$_A$
5	Inputs: 3[1], 4[1] Outputs: 11.0 → addr$_B$
6	Inputs: 3[0], 5[0] Outputs: 3.0 → addr$_D$, 7.0 → addr$_C$, 9.0 → addr$_B$

TABLE I: Transaction examples.

Payments are grouped by transactions (see Table I for a hypothetical toy example), and the transactions are recorded into a public database called the blockchain. The content of an actual Bitcoin transaction comprises metadata, inputs and outputs [16]. The metadata includes the hash of the entire transaction, which is a unique ID for the transaction (in Table I the transactions are labelled from 1 to 6 instead). Every input specifies a previous transaction, identified by its hash, and the index of the previous transaction's output that is being spent. For example, the inputs of transaction 5 are 3[1] and 4[1], referring to the second outputs (outputs are labelled from 0) of transactions 3 and 4. Since the same address is used, we see this transaction is a fund consolidation. Every output contains the value (amount) of the transaction, and a mechanism to identify the recipient address.

Therefore the Bitcoin network can be modelled as a heterogeneous graph, where nodes comprise transactions, identified by their unique hash, and Bitcoin addresses, typically a user may own many Bitcoin addresses. From each input of a given transaction T, a directed edge is created, linking an output of some previous transaction T' to T. This same edge will also appear as an output of T', from which the transaction amount can be read. This model will justify the adoption of the graph database Neo4j as part of our system architecture described in Subsection III-A.

Different networks can be extracted (shown in Figure 1) from the Bitcoin network: (1) A transaction only network, as used e.g. in [11], represents the flow of Bitcoins between transactions over time. Each vertex represents a transaction, and each directed edge between a source vertex to a target vertex connects an output transaction of the source, to the target where it is used as input, as illustrated on Figure 1a: we have 6 transactions, thus 6 nodes. The first transaction has no input, but one output, transaction 2 has one input and two outputs, each of them are inputs to respectively transactions 3 and 4. (2) An input-output network can be considered, where nodes are addresses, and there is a directed edge when an output serves as input to a transaction. This is possible without ambiguity if a transaction has one input with possibly several outputs (as in transaction 3 in our example), or possibly several inputs and one output (as in transaction 5 in our example), but not when there are multiple input multiple output transactions (as in transaction 6, see Figure 1b), since then we only know the inputs and outputs of the transaction node, but not how they relate to each other. When there are two outputs, oftentimes one of the outputs is for collecting change, while the other is the intended payee of the transaction (though it is still not clear which is which, and someone simply consolidating funds may also create two outputs as a ruse). This observation is the idea behind mixing, which obfuscate individual pairings in multi-input multi-output transactions. (3) A (Bitcoin wallet) address network could be created following e.g. [15], where each node represents a Bitcoin (wallet) address, and there is a directed link between two nodes if there was at least one transaction between the corresponding addresses. Wallet addresses are sometimes aggregated via some heuristic to create a user network, as in e.g. [11].

III. BiVA Design

A. System Architecture

The Bitcoin blockchain data was downloaded with Bitcoin Core, and parsed to be stored in the Neo4j (v3.4.1) [21] graph database management system, which forms the back-end of BiVA. The graph model we consider is heterogeneous and consists of two types of nodes (2-mode network), transactions and (wallet) addresses, which we model using two different labels in Neo4j. Relationships are also modelled as two types, depending on whether a directed edge describes an input to or an output from a transaction. This data schema allows us to derive the three network representations described in Fig. 1. This is illustrated in Fig. 4: the upper part shows the 2-mode network, obtained as the neighborhood of a chosen query (wallet) address within a perimeter of 2. Pink nodes are transaction nodes, and the query address is shown to be

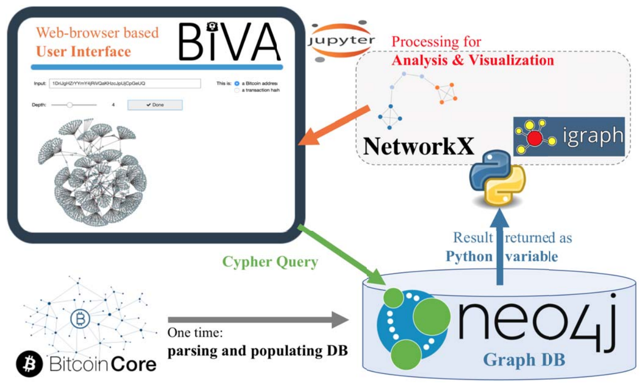


Fig. 2: BiVA system architecture

both an output and an input to two respective transactions. The lower part contains the corresponding address network, obtained by removing the transaction nodes. Since we have multi-input and output transactions, the directed link going from the query address to the transaction is changed into two links to both transaction outputs. Similarly, the incoming edge to the query address becomes two incoming edges from addresses. Fig. 5 shows the corresponding address record as obtained from blockexplorer.com.

The front-end of BiVA is implemented in Python. We use Jupyter dashboard [25] to realize a web-browser based front-end, which embeds the codes necessary for querying and processing that can be easily hidden, to expose only the dashboard based user interface. Snippet of the interface so realized is shown in Figure 3. The back-end database is queried with the Cypher query language [22] using ipython-cypher, and the returned result is passed as a Python variable for further processing. This includes execution of graph algorithms (standard ones, as well as custom algorithms designed for BiVA, as described in Section III-B) that leverage on the NetworkX [23] package for analysis, and the igraph [24] package for graph rendition.

This modular architecture allows separation of the data storage in Neo4j, the data processing using Python and NetworkX, and the user interaction and exposition of the results. It provides a clean web-browser based interface for users interested in using BiVA as it is, yet empower the user to fully control, check and even modify the implementation still using the same browser based interface, to adapt and add any other desired functionalities.

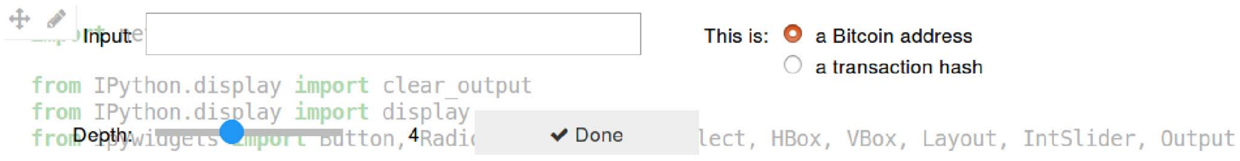
B. *BiVA's Underlying Graph Algorithms*

We next elaborate the new graph analytics algorithms used in BiVA. Since we will illustrate them on the address network, in the following we will denote by $G = (V, E)$ the graph formed by having Bitcoin addresses as vertices in V, and directed edges in E representing a flow of Bitcoins from one address to another (using the convention explained earlier for multi-input multi-output transactions). We use the notation $\overleftrightarrow{G} = (V, \overleftrightarrow{E})$ to refer to an undirected graph obtained from G, such that $\forall(u, v) \in E$, $(u, v) \in \overleftrightarrow{E}$ and $(v, u) \in \overleftrightarrow{E}$. Likewise, we use the notation $\overleftarrow{G} = (V, \overleftarrow{E})$ to denote the directed graph obtained by reversing all the edge directions from G, so to say, $(v, u) \in \overleftarrow{E} \iff (u, v) \in E$. Due to space limitation, below, we omit pseudocode descriptions, and instead provide the intuition of how the algorithms are realized.

BiVA provides two novel information theory based algorithms, that derive *macroscopic* information from the address graph - one for node centrality and one for clustering.

Entropic centrality: Node centrality is a measure of the importance of a node in a graph. There are numerous ways to define node centrality [26], the most well known being arguably the degree centrality, where a node is important if its degree is high. In [27], the notion of entropic centrality was proposed, to formalize that a node $v \in V$ is central if, given a flow that starts at v, there is a high uncertainty about the destination of this flow. Entropy is a well known information theoretic measure of uncertainty, and the entropic centrality is thus the entropy attached to the probability of a flow reaching any node via a path given that it started at v. The condition that the flow follows a path was changed in [28] by considering a Markov model, where the flow was replaced by a random walker starting at v, and reaching other nodes with a probability inversely proportional to the number of neighbors a node has. The walker walks for t times, and there is an entropic centrality $C^t_H(v)$ for every choice of vertex v and time t. The Markov entropic centrality was used in [28] as an ingredient for a clustering algorithm. We revisit the concept of Markov entropic centrality as a centrality measure per se, and implement an entropic centrality algorithm for asymptotic t, to capture the overall influence of v over the network, considered as a directed graph. The motivation for considering this approach in the Bitcoin network context is to capture the flow of Bitcoins, and in particular, determine how influential a particular wallet address is in terms of Bitcoin propagation in the address network G. We also consider the reversed address network $\overleftarrow{G}$ when the interest is to determine how important an address is in terms of receiving Bitcoins (that is, being the destination of a random walker starting at any other address in the address network).

Clustering based on confined circulation of probabilistic flows: Encapsulating Bitcoin flows is also behind the main idea of a novel clustering algorithm. Many clustering algorithms for directed graphs rely on known algorithms for undirected graphs, as classified in [29], where it was also noted that one of the approaches customized for directed graphs consists in identifying clusters where flow in the graph gets trapped, e.g. [30], [31]. Our approach falls in this category: we start by finding nodes with low entropic centralities, and then look at the probability to visit neighbors of such nodes, and cluster these probabilities, using AgglomerativeClustering (Euclidean distance, ward linkage) from Scikit-learn [32] on the probability values. This identifies neighbors of low entropic nodes where the flow is most likely to go, in a sense identifying local communities. This approach appears to be well suited to



Input:

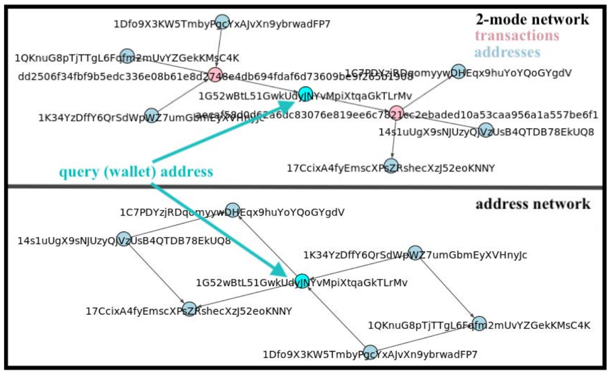
This is: [✓ a Bitcoin address] a transaction hash

Depth: 4

✔ Done

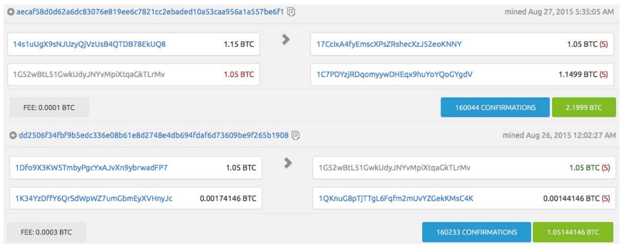
```
from IPython.display import clear_output
from IPython.display import display
from ipywidgets import Button, Radio                   lect, HBox, VBox, Layout, IntSlider, Output
```

Fig. 3: Jupyter dashboard based user & programmer interface snippet.



2-mode network
transactions
addresses
query (wallet) address
address network
1Dfo9X3KW5TmbyPgcYxAJvXn9ybrwadFP7
1QKnuG8pTjTTgL6Fqfm2mUvYZGekKMsC4K
dd2506f34fbf9b5edc336e08b61e8d2748e4db694fdaf6d73609be9f265b1908
1C7PDYzjRDqomyywDHEqx9huYoYQoGYgdV
1G52wBtL51GwkUdyJNYvMpiXtqaGkTLrMv
1K34YzDffY6QrSdWpWZ7umGbmEyXVHnyJc
aecaf58d0d62a6dc83076e819ee6c7821cc2ebaded10a53caa956a1a557be6f1
14s1uUgX9sNJUzyQjVzUsB4QTDB78EkUQ8
17CcixA4fyEmscXPsZRshecXzJ52eoKNNY

Fig. 4: An ego-centric view of a 2-mode (transaction-address) network, and the corresponding address network.



aecaf58d0d62a6dc83076e819ee6c7821cc2ebaded10a53caa956a1a557be6f1 mined Aug 27, 2015 5:35:05 AM

14s1uUgX9sNJUzyQjVzUsB4QTDB78EkUQ8	1.15 BTC	17CcixA4fyEmscXPsZRshecXzJ52eoKNNY	1.05 BTC (S)
1G52wBtL51GwkUdyJNYvMpiXtqaGkTLrMv	1.05 BTC	1C7PDYzjRDqomyywDHEqx9huYoYQoGYgdV	1.1499 BTC (S)

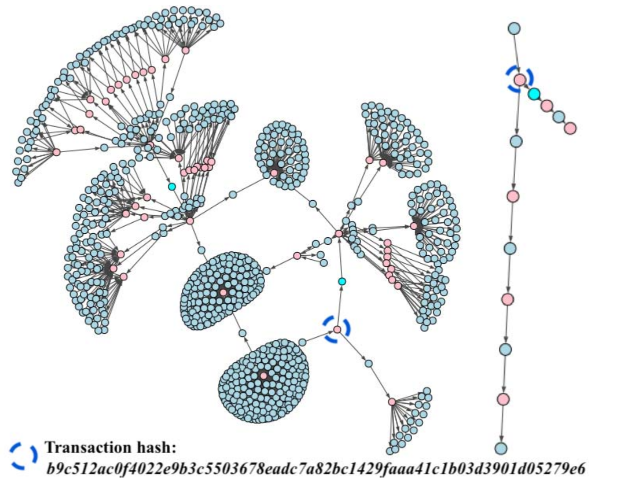
FEE: 0.0001 BTC 160044 CONFIRMATIONS 2.1999 BTC

dd2506f34fbf9b5edc336e08b61e8d2748e4db694fdaf6d73609be9f265b1908 mined Aug 26, 2015 12:02:27 AM

1Dfo9X3KW5TmbyPgcYxAJvXn9ybrwadFP7	1.05 BTC	1G52wBtL51GwkUdyJNYvMpiXtqaGkTLrMv	1.05 BTC (S)
1K34YzDffY6QrSdWpWZ7umGbmEyXVHnyJc	0.00174146 BTC	1QKnuG8pTjTTgL6Fqfm2mUvYZGekKMsC4K	0.00144146 BTC (S)

FEE: 0.0003 BTC 160233 CONFIRMATIONS 1.05144146 BTC

Fig. 5: Record (from https://blockexplorer.com/) for the wallet address *1G52wBtL51GwkUdyJNYvMpiXtqaGkTLrMv*.



Transaction hash:
b9c512ac0f4022e9b3c5503678eadc7a82bc1429faaa41c1b03d3901d05279e6

Fig. 6: Filtering of a Bitcoin subgraph, shown on the left, that detects a chain of large ($\geq$ 2000฿) transactions, on the right.

capture the probabilistic flow of Bitcoins because of mixing in the directed address network, where multi-input multi-output transactions are modelled by assuming that the Bitcoin flow goes from every incoming address to every outgoing one.

Next, we discuss two *microscopic* (ego-centric) graph algorithms which are centered around a pair/set of address nodes (in contrast to the above described centrality and clustering algorithms which are macroscopic). These algorithms are applied when a BiVA user has identified certain addresses of interest - either by some out-of-system mechanism, or during the exploratory process using BiVA, for instance, to zoom into a subset of nodes within a cluster identified as above, or study certain high centrality nodes.

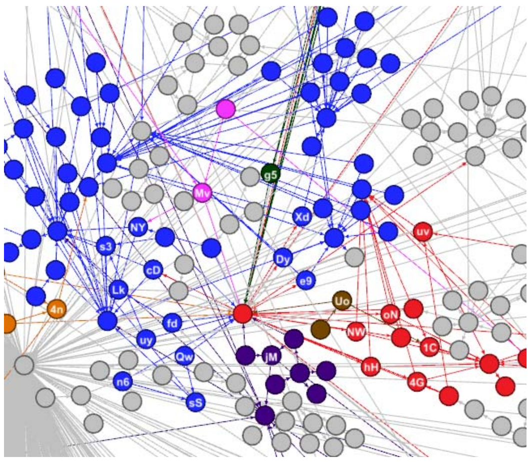
Path confluence based address aggregation: Consider a (typically shortest) path P between two vertices v_1 and v_2 in the undirected graph $\overleftrightarrow{G}$, which includes a vertex v_x such that this path in $\overleftrightarrow{G}$ is in fact composed of two paths v_1 to v_x and v_2 to v_x (or conversely from v_x to each of v_1 and v_2) in G, then we try to identify another vertex v_y (not already contained in the original path P) in the graph G such that there exist disjoint paths from v_y to both v_1 and v_2 (or conversely, from both v_1 and v_2 to v_y). Doing so, we identify two disjoint confluencing paths in G between v_x and v_y, such that one of these paths contains v_1 and the other contains v_2. Such confluence, particularly if the path lengths (between v_x and v_y) are not very long, are statistically unlikely unless all the nodes on the paths are in fact acting in a coordinated manner.

The path confluence query is achieved by tweaking the breadth first search (BFS) algorithm, starting the BFS with the original group of nodes in P marked as visited, and then subsequently identifying as v_y whichever new (not in P) node gets revisited during the BFS process.

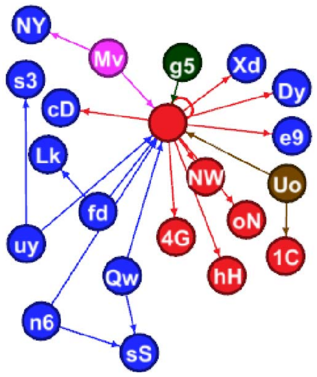
Bitcoin flow traceback/traceforward: Given a set S of address nodes, we may want to trace where they collectively get the money from. Accordingly we identify the set $S_x \subseteq V$ that has any path to at least some node in S. S_x can be identified by traversing the edges in the reverse sense using a breadth first search (BFS). In terms of practicality, it may also make sense to limit the search within a distance threshold instead of exploring the whole graph. Next, we consider the graph G' which comprises only the vertices that are in $S \bigcup S_x$, and the edges that are among nodes in this subset. We consider that each edge $e \in E$ has a weight, determined by the Bitcoin amount. Consider now the case of a multi-input multi-output transaction, where an incoming address x_i has an edge of weight w_i to the transaction, and the transaction has directed edges with weight w_o to outgoing address y_o. Then, in the

1C	14YeaK34GE68S6YASdtHR1iThj8c8hBQ1C
4n	1Gscvx3ugexW4xKjBFogQtQRsHR9Eh8v4n
4G	1MAwvr21q95UBu6y4WhRM47K5RBijQ164G
cD	1EuRpZCEoyRKWcEpv6jD42N2CxJWrrETcD
Dy	12bbs3MZAsd9mseXKTA93PtHVd2nSkvjDy
e9	1Hmjw6JcNDxCQuVo7pkUmRmJzRJe4yLze9
fd	1ErwF3T6QCdZ75PBXZdyxXJPN2k7bBV9fd
g5	1JBygewRQuHuh4qJnpQtb1qCfwNJ2zNrg5
hH	1Kbj3tzAbtHgJHt6h6G5PuvWYRZkWXAfhH
jM	1LnxsZDPNkU96mrYoS1CQwxPmrhJyrgujM
Lk	1HZErPx5XPrp8mH98rKRjyazQLYN7j34Lk
Mv	1G52wBtL51GwkUdyJNYvMpiXtqaGkTLrMv
n6	18euqRRpC2Zp9i9dwrT7Qp3M8jfbu9TUn6
NW	1KFc1tekeQtZg48pKbCEU2j7J14N2XFTNW
NY	17CcixA4fyEmscXPsZRshecXzJ52eoKNNY
oN	13Mj4nmV127symEJ6GTTb4kdTWyQ884QoN
Qw	1Q3AvrmZ8cykHGQ7kCkSZaopnWiHPzc1Qw
s3	1PUt7bm1ZU6BRT85yDJxd651Fgneveuzs3
sS	14gkHySfCnoV4Fy4nCxoLUgavMP3DMXKsS
uv	1ETmDWjyto3gGegqch41PCeYJiVx9pmsuv
uy	1Nr2y7XD8c27tMQF5XrQJdfUWjCZ3zY5uy
Uo	1BkUNiTfJRTTG1iycaWGZavmLnLVHorUUo
Xd	137ENC6fNM97tawPMAGgyn9BKLEmkTx7Xd

(a) Suspect address index.



(b) Probabilistic flow based clustering.



(c) Connections among close-knit suspect addresses.

Fig. 7: Identifying clusters and zooming into suspect addresses.

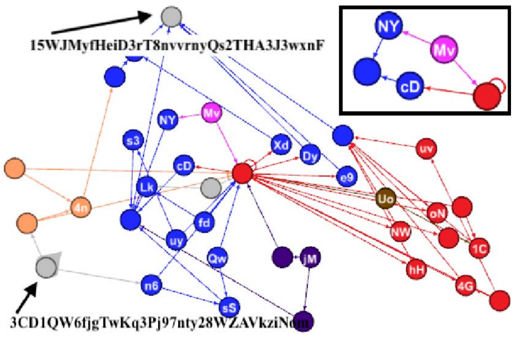

15WJMyfHeiD3rT8nvvrnyQs2THA3J3wxnF
NY
Mv
cD
s3
Xd
Dy
uv
e9
Lk
4n
Uo
oN
fd
NW
1C
uy
Qw
jM
hH
n6
sS
4G
3CD1QW6fjgTwKq3Pj97nty28WZAVkziNdm

Fig. 8: Addresses aggregated with path confluences

address graph, x_i will have a directed link to y_o with weight $w_i(w_o/\sum_i w_i)$. This weighted average acts as a proxy, in the absence of determinism due to mixing. The expectation is that even if one such a guess instance is just noise, aggregated volume can be used to draw attention to nodes of interest. We thus consider the sum of the weights of incoming edges to a vertex in G' as an indicator of the expected amount of money flowing through it to the original set of nodes of interest S. The same idea can be applied on $\overleftarrow{G}$ to deduce where the money from these nodes S flow to downstream.

Some results obtained with the above algorithms will be illustrated next. Since the purpose of the demo is to expose BiVA as a tool for analyzing and visualizing the Bitcoin network, more rigorous investigations of the underlying algorithms are deferred to future work.

IV. Demo

A. Data Set Exploration

BiVA supports the visualization of standard queries on the Bitcoin network: display of the 2-mode network (or of the transaction only network, or of the address network) centered around a specific wallet address or transaction query within a given parameter or time frame, display of shortest/multiple paths among addresses on the 2-mode network or the address network, to name a few. BiVA can also filter the edges by the amount of Bitcoins involved, or find addresses of interest (for instance, addresses with high centrality, addresses that are aggregated together based on several existing and newly proposed algorithms) in the neighborhood of the specific query address. In Figure 6, we show a 2-mode network, centered around the address 1HxuYkYf9SXAD9raVV3UKe3AZvrREgviYs. This node, in light blue, appears twice, and is involved in 4 transactions (shown in pink). By filtering the network shown on the left, looking for transactions of more than 2000Ƀ, we identify a chain of large transaction amounts, shown on the right.

We next focus on results from some of the new algorithms, by exploring an Ashley Madison data breach extortion.

B. Case study: Extortion of Ashley Madison breach victims

To continue the exposition of some of the features of BiVA, we consider a blackmail extortion campaign (from 23 August 2015 [1]) targeting Ashley Madison data breach victims. From the blog [1] discussing this incident, we collected 120 Bitcoin addresses (67 identified by the blogger, the rest was reported in the blog comments). The extortion emails sought a precise amount of 1.05Ƀ, and had a deadline of 3 days from the demand. The 67 addresses received the said amount in the relevant time frame. This gave us an initial shortlist of suspect addresses, not all of which were necessarily involved in the extortion, while the initial suspect list also does not capture many other addresses the extortionists control to move the extorted money around.

In Figure 7b[1], we report our clustering result on a directed address subgraph containing nodes within a dis-

[1]BiVA relies on igraph for visualization, however the figures in this section were redone using Gephi [33] to edit the figures so as to fit the paper format.

tance of 5 (in an undirected variant of the address graph) centered around one of the suspect addresses, namely, *1G52wBtL51GwkUdyJNYvMpiXtqaGkTLrMv*. This subgraph had 4571 nodes and 4762 directed edges, and 23 of the aforementioned suspected nodes[2]. Our probabilistic flow based clustering algorithm is implemented in a bottom-up agglomerative manner, and we show the results from an intermediate fifth iteration of agglomeration (we focus on the portion of the subgraph containing the 23 suspect addresses post clustering), just before the two (red and blue) clusters (with largest number of suspect nodes) of interest coalesce. The nodes with labels indicate the originally suspect addresses. Nodes in clusters with no originally suspected address are all shown in gray. 12 suspect nodes are in the blue cluster comprising 120 nodes (11 of these 12 were clustered together already in iteration 1, which at that point had 24 nodes), while the red cluster of 15 nodes contains 6 of the suspected nodes. The unsupervised aggregation of some of the suspected addresses by our flow based clustering is both an initial validation of the clustering approach itself; conversely, the other members of these clusters, even though not originally suspected, become new addresses of interest.

We next zoom into the close knit suspect addresses (Figure 7c), and apply the path confluence based address aggregation mechanism among some pairs among them. The inset of Figure 8 shows the path confluence mechanism in action when a single pair of addresses (index NY and cD) are considered, while a superimposed result from multiple path confluences is shown as the main part of Figure 8.

The bottom gray node identified in Figure 8 has the wallet address of *3CD1QW6fjgTwKq3Pj97nty28WZAVkziNom*, which, as of 15 July 2018, participated in 1105338 transactions amounting 3517872.12188175Ƀ, and was mentioned in several forums discussing diverse scams. The top gray node represents *15WJMyfHeiD3rT8nvvrnyQs2THA3J3wxnF* which collected 8.25Ƀ from a single transaction (with only one other output of 0.01239939Ƀ), involving several of the already suspected nodes, as well as other nodes (many were already identified to be in the same cluster as the suspected nodes, which is another indicator that the original flow based clustering algorithm as well as the path confluence based address aggregations yield meaningful results). For a forensic investigator, these newly identified addresses thus become nodes of interest.

Due to space limitation, we omit screenshots of the results obtained through traceback/forward algorithms, which similarly identify (many common) new suspect nodes.

Source code is available at: https://github.com/feog/Biva

References

[1] https://blog.cloudmark.com/2015/09/01/does-blackmailing-pay-signs-in-bitcoin-blockchain-of-responses-to-ashley-madison-extortion-emails/, accessed on 14 July 2018.

[2] B. Zheng et al., "Malicious Bitcoin Transaction Tracing Using Incidence Relation Clustering", in "Mobile Networks and Management", (MONAMI) 2017.

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

[4] M. Spagnuolo, M. Federico, Z. Stefano, "Bitiodine: Extracting intelligence from the bitcoin network", in "Intl. Conf. on Financial Cryptography & Data Security" 2014.

[5] M. Lischke and B. Fabian, "Analyzing the Bitcoin Network: The First Four Years", in "Future Internet", 8(7) 2016.

[6] M. Harrigan, C. Fretter, "The Unreasonable Effectiveness of Address Clustering", in "arXiv", 2016.

[7] K. Liao, Z. Zhao, A. Doupe, G-J. Ahn, "Behind closed doors: measurement and analysis of CryptoLocker ransoms in Bitcoin", in "APWG Symposium on Electronic Crime Research (eCrime)", 2016.

[8] H. Kuzuno, C. Karam, "Blockchain Explorer: An Analytical Process and Investigation Environment for Bitcoin", in "APWG Symposium on Electronic Crime Research (eCrime)", 2017.

[9] E. Androulaki, et al. "Evaluating user privacy in bitcoin", in "Intl. Conf. on Financial Cryptography and Data Security" 2013.

[10] J.D. Nick, "Data-Driven De-Anonymization in Bitcoin", in "ETH Master Thesis", 2015.

[11] F. Reid, M. Harrigan, "An Analysis of Anonymity in the Bitcoin System", *Altshuler Y., Elovici Y., Cremers A., Aharony N., Pentland A. (eds), Security and Privacy in Social Networks. Springer*, New York, 2013. https://arxiv.org/pdf/1107.4524.pdf

[12] M. Moser, R. Bohme, "Anonymous Alone? Measuring Bitcoin's Second-Generation Anonymization Techniques", in "IEEE European Symposium on Security and Privacy Workshops", 2017.

[13] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. Kroll, E.W. Felten, "SoK: Research perspectives and challenges for bitcoin and cryptocurrencies", in "IEEE Symposium on Security and Privacy", *S&P* 2015.

[14] D. Ermilov, M. Panov, Y. Yanovich, "Automatic Bitcoin Address Clustering", in "IEEE Intl. Conf. on Machine Learning and Applications", *ICMLA* 2017.

[15] 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(2): e86197.

[16] A. Narayanan, J. Bonneau, E. Felten, A. Miller, S. Goldfeder, "Bitcoin and Cryptocurrency Technologies", *Princeton University Press*, 2016.

[17] M. Bartoletti, S. Lande, L. Pompianu, A. Bracciali, "A general framework for blockchain analytics", in "Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers", 2017.

[18] S. Bistarelli, F. Santini, "Go with the -Bitcoin- Flow, with Visual Analytics", in "Intl. Conf. on Availability, Reliability & Security", *ARES* 2017.

[19] https://blockchain.info, accessed on 12 July 2018.

[20] https://blockexplorer.com, accessed on 12 July 2018.

[21] https://neo4j.com, accessed on 12 July 2018.

[22] N Francis et al., "Cypher: An Evolving Query Language for Property Graphs", in "Intl. Conf, on Management of Data" 2018.

[23] A.A. Hagberg, D.A. Schult and P.J. Swart, "Exploring network structure, dynamics, and function using NetworkX", in 'Python in Science Conference", (SciPy) 2008.

[24] G. Csardi and T. Nepusz, "The igraph software package for complex network research", in 'InterJournal", 2006.

[25] https://jupyter-dashboards-layout.readthedocs.io/en/latest/, accessed on 12 July 2018.

[26] E. Estrada, "Centrality measures", in "The Structure of Complex Networks: Theory and Applications", *Oxford University Press*, 2011.

[27] F. Tutzauer, "Entropy as a measure of centrality in networks characterized by path-transfer flow", *Social Networks*, 29, 2007.

[28] A.G. Nikolaev, R. Razib, A. Kucheriya, "On efficient use of entropy centrality for social network analysis and community detection", *Social Networks*, 40, 2015.

[29] F. D. Malliaros, M. Vazirgiannis, "Clustering and Community Detection in Directed Networks: A Survey", *Physics Reports*, 533(4), 2013.

[30] M. Rosvall, C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure", *Proc. of the National Academy of Sciences*, 105 (4), 2008.

[31] Y. Kim, S.-W. Son, H. Jeong, "Finding communities in directed networks", *Physics Review E*, 81, 016103, 2010.

[32] Pedregosa et al., "Scikit-learn: Machine Learning in Python" Journal of Machine Learning Research, Vol. 12, 2011.

[33] M. Bastian, S. Heymann, M. Jacomy, "Gephi: an open source software for exploring and manipulating networks", *AAAI Conf. on Weblogs and Social Media*, 2009.

[2]Overall, 61 of the suspected nodes were in a single connected component of the undirected variant of the address graph.