Go with the -Bitcoin- Flow, with Visual Analytics*

Stefano Bistarelli
Dip. Matematica e Informatica, University of Perugia
Via Vanvitelli 1, 06123, Perugia
stefano.bistarelli@unipg.it

Francesco Santini
Dip. Matematica e Informatica, University of Perugia
Via Vanvitelli 1, 06123, Perugia
francesco.santini@unipg.it

ABSTRACT

Bitcoin is a cryptocurrency and a peer-to-peer payment system, where transactions directly take place between pseudo-anonymous users, without any centralised authority. Since the block-chain (i.e., the public ledger where transactions are registered) is an example of Big Data, a straightforward visualisation is not very informative. For this reason, we employ techniques from Visual Analytics to filter out undesired information in order to obtain a tool to visually analyse the transactions and help its analysis. For instance, different views can highlight miners, or sources and leaves of bitcoin flows, together with the balance of each address and transaction. Moreover, the main view sees transactions as grouped into disconnected "islands", making it possible to focus on only one of them at once.

CCS CONCEPTS

• **Information systems:** Digital cash; • **Human-centered computing:** ***Visualization toolkits***; ***Visual analytics***;

KEYWORDS

Bitcoin, Visual Analytics, Analysis Tool.

ACM Reference format:
Stefano Bistarelli and Francesco Santini. 2017. Go with the -Bitcoin- Flow, with Visual Analytics. In *Proceedings of ARES '17, Reggio Calabria, Italy, August 29-September 01, 2017,* 6 pages.
https://doi.org/http://dx.doi.org/10.1145/3098954.3098972

1 INTRODUCTION

In this work we describe *BlockChainVis*[1], a tool dedicated to the visual analysis of flows of Bitcoin transactions [13]. Since the block-chain is an example of Big Data, a straightforward visualisation in its entirety is not significant. Hence, we have exploited some techniques from *Visual Analytics* (*VA*) [19] to filter out undesired information and focus on some characteristics.

Criminals often find ways to exploit legitimate technologies for nefarious uses. Bitcoin is not different: due to its pseudo-anonymity and untraceability, it is currently used for criminal activities such as money laundering, and selling illegal goods or services. Bitcoin has been the de-facto currency of the Dark Web, the "hidden" Internet accessible only by *Tor*[2], since the pioneering marketplace *Silk Road*, also known as the "eBay of drugs", arrived in 2011. After its shutdown, new *darknet markets* proliferated: these digital markets primarily are black markets, selling or brokering transactions involving drugs, cyber-arms, weapons, counterfeit currency, stolen credit card details, forged documents, unlicensed pharmaceuticals, steroids, other illicit goods, as well as fully-legal products.

Laundering money using bitcoin is attractive[3] because of its low fees, instantaneous transactions, and virtually anonymous. For instance, bitcoins collected from illegal activities, e.g., proceeds of previously-mentioned markets, can be mixed with "clean" money by using *mixing services* (also called *tumblers*). These services[4] are third parties used to break the connection between a Bitcoin address sending coins and the address(es) they are sent to. Basically, the destination address of every mixed transaction receives the same amount of money from possibly many different addresses.

The aim of BlockChainVis is to help analysing all such scenarios in deep. At a first step, the tool highlights transaction islands, i.e., the sub-graphs disconnected from the super-graph, which represents the whole block-chain. Then it is possible to apply further filters on, e.g., the interval of dates, blocks, transaction values, or number of addresses used in transactions, using VA techniques. Such views highlight specific nodes (e.g., roots and leaves of flows, or miners) and exclude useless and confusing information: e.g., transactions representing changes can be hidden.

The paper is organised as follows: in Sec. 2 we describe the Bitcoin peer-to-peer network; then, Sec. 3 shows architecture of BlockChainVis and the technologies behind it. In Sec. 4 provides details on the VA techniques used to visualise only needed information.[5] Section 5 shows a case-study from the real-world, based on identifying a mixing-service. We conclude with related work (Sec. 6), and then conclusions and future work (Sec. 7).

2 BITCOIN

The white-paper on Bitcoin appeared in November 2008 [13], under the pseudonym "Satoshi Nakamoto". His invention is an open-source, peer-to-peer digital currency. Money transactions do not require a third-party intermediary, with no traditional financial-institution involved (฿ is the commonly used currency-symbol).

The payer and payee directly interact (peer-to-peer), but without using their real identities, and no personal information is transferred

*Research supported by "Fondazione Cassa di Risparmio di Perugia" (Ref. Bitcoin).
[1] http://www.dmi.unipg.it/blockchainvis/.

ARES '17, August 29-September 01, 2017, Reggio Calabria, Italy

ACM ISBN 978-1-4503-5257-4/17/08…$$15.00
https://doi.org/http://dx.doi.org/10.1145/3098954.3098972

[2] Tor project: https://www.torproject.org.
[3] The Guardian: http://tinyurl.com/hmp2ll7.
[4] E.g., *Bitmixer*: https://bitmixer.io.
[5] Note that in the following text we suppose the paper will be printed in greyscale, thus we do not explicitly refer to colours in the pictures, but e.g., we use the adjectives "darker" or "lighter" to distinguish.

from one to the other. However, unlike a fully anonymous transaction, there is a transaction record. A complete transaction record of every bitcoin and every Bitcoin user's encrypted identity is maintained on a public ledger, called the *block-chain*. For this reason, Bitcoin transactions are thought to be *pseudonymous*, not anonymous: Bitcoin addresses are pseudonyms of real individuals (one can have several pseudonyms).

The only way to create new bitcoins is through the *mining* process: *miners* are the nodes that verify the transactions and add them to the block-chain. The number of bitcoins created each time a miner discovers a new block represents a reward for its job.

The actors in the Bitcoin network are the *users* who own a *wallet* associated with a couple (or more) of private/public cryptographic keys. In Bitcoin, a private key is usually a 256 bit random number, and by using the *Elliptic Curve Digital Signature Algorithm* (*ECDSA*) [9], a 512 bit public key can be obtained from it. Afterwards, from the public key it is possible to obtain a Bitcoin *address*, e.g., applying an hashing function on it. Users use these keys to sign the transactions they generate in order to transfer their money to other users; transactions are then broadcast to the Bitcoin peer-to-peer network. The miners update the block-chain, a database of every transaction ever executed.

Transactions. Transactions are the basic brick of the Bitcoin network (and of our visualisation): they represent the mechanism that allows a user to cede money to another user, e.g., from a buyer to a seller. The new owner can prepare a new transaction referring to the ones through which she received money, called the (multiple) *inputs* of this new transaction. The *output* of a transaction describes the destination of bitcoins instead. There can be multiple *outputs*, allowing a owner to make multiple payments at once; one output often represents the change w.r.t. a previous transaction. Finally, a user broadcasts her transaction to the Bitcoin network.

Block-chain. Miners keep the block-chain consistent, complete, and unalterable: they repeatedly verify and collect newly broadcast transactions into a new group of transactions, called a *block*. Mining is used to introduce bitcoins into the system, due to a reward (now 13฿) created and assigned to the winning miner. This serves the purpose of disseminating new coins in a decentralised manner, as well as motivating people to provide security for the system.

The first step accomplished by a miner, after collecting transactions, is to perform a verification step on them. This implies to check a set of rules, e.g., if their format is syntactically correct w.r.t to the protocol, or to reject it if the sum of input values is less than sum of output values. Valid transactions (all checks are passed) are added to a block. A block consists of a header and a list of transactions (the block body). Each block contains information that chains it to the previous block in the block-chain, that is the hash of the previous block. Thanks to this field, a block (and consequently the block-chain) is computationally impractical to be modified, since every block after it would also have to be regenerated. The remaining field of the header, i.e., the *nonce*, is obtained from the computation of the proof-of-work by miners.

Simply speaking, with such a calculation a miner aims at finding a random *nonce* (a little random data) that becomes part of a block and makes it have a hash that starts with a given amount of zeroes. This nonce corresponds to a 32-bit field that, once inserted into the current block-header, makes its hash be less in value than the current *target*. The target is a 256-bit number (extremely large) that all Bitcoin nodes share. The most widely scheme for hashing is *SHA256* [8]. This proof is easy to verify, but extremely time-consuming to generate: the demanded work is exponential in the number of "zero bits" required in the head of the block-hash, but it can be verified in a single hash.

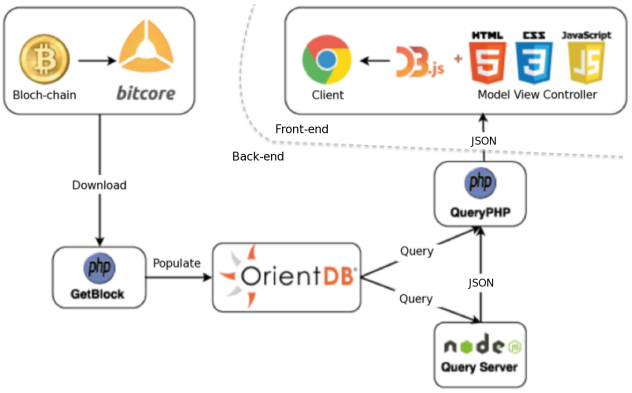

Figure 1: A summary of the technoogies used in the front/back-end.

3 ARCHITECTURE AND TECHNOLOGIES

BlockChainVis is a client-server Web-application. It consists of a back-end (server-side) and a front-end (client-side). The main technologies used for the back-end are *OrientDB*[6], *PHP*, *Node.js*[7], and *Bitcore*[8], while the ones for the front-end are *HTML5*, *CSS3*, *Javascript*, and *D3.js*[9] (see Fig. 1).

The back-end of BlockChainVis is implemented on a machine with 128Gbyte of RAM, 2 processors Intel(R) Xeon(R) CPU E5-2620 v4 2.10GHz 8 core (for a total of 32 threads); in particular, the implementation consists of three different virtual machines running, *i)* Bitcore, *ii)* OrientDB, and *iii)* software dedicated to visualisation.

As a first step, the tool downloads the entire block-chain in the back-end; in fact, to intensively work on it, after some initial attempts we discarded the idea to use a *block-explorer* because of their current limitations (traffic volumes and omission of some information). Block-explorers are Web-sites that allows for reviewing information about the block-chain, by using dedicated Web-services. Therefore, we opted for *Bitcore*, which is a *full*[10] Bitcoin-client. Bitcore block-chain can be queried by using *Insight API*, and the result is presented as a *JavaScript Object Notation* (*JSON*[11]) file, which is a simple text-document where the basic structure is a set of name-value pairs and an ordered list of values.

The second step consists in extracting the desired information from the block-chain in order to populate the database. This can be done directly from the front-end, by selecting *Settings* (using Admin

[6] http://orientdb.com/orientdb/.
[7] https://nodejs.org/en/.
[8] https://bitcore.io/.
[9] https://github.com/aaronpowell/db.js/.
[10] Full nodes download every block in the block-chain.
[11] JSON: http://www.json.org.

credentials) in the opening Web-page of BlockChainVis (the only further selection is *Visualisation*). As the (graph) database we use *OrientDB* (see Sec. 3.1). The PHP script that accomplishes this task is *getBlock.php*, which takes as input a range of blocks. Starting from the first block, by calling the API offered by BitCore, the script extracts all the information related to it (the result of the invocation is a JSON file) and encodes it in a PHP object to better handle it. Then, the *Coinbase* (the first transaction in a block, which generates new bitcoins as reward) and all the other transactions are added to the database. At the end of the scan, the script generates a report message and switches to the next block.

PHP scripts are also used to operate on the OrientDB database in order to serve requests from the client application: the extracted information is then translated to JSON. In addition, for some of the queries we exploit the OrientDB driver for Node.js and Java.

The design-pattern we have adopted to manage the user-interface is the classical *Model-view-controller* (*MVC*) [10]. Such an architecture divides an application into three interconnected parts, with the purpose to separate the internal representation of information from the ways it is presented to a user. The *Model* manages the data, logic, and rules of the application. In Sec. 4 we provide some examples of functions implemented in each of these three parts.

3.1 The Database

OrientDB is an open source *NoSQL* ("Not only SQL") database management system written in Java. It is a multi-model database, supporting graph (the only model we use), document, key/value, and object models, but the relationships are managed as in graph databases with direct connections between records. Compared with relational databases, graph databases scale more naturally to large data sets, as they do not typically require expensive join operations. Since they are less dependent on a rigid schema, they are more suitable to manage ad-hoc and changing data with evolving schemas.

The four classes to represent all the stored information are:

- The *Transaction* class is a sub-class of *V*ertex (the native class to represent nodes). The attributes of Transaction are *Hash* (the hash of the transaction is its primary key), *Height* (the height w.r.t the block the transaction is recorded), *Miner* (it describes whether the transaction has generated new bitcoins), *Time* (date and time it has been recorded), *TX_Fee* (the total paid fee), and *ValueTot* (the sum of all the inputs).
- The *Address* class is, as Transaction, a sub-class of Vertex: these are the only two kinds of nodes in the DB. The only attribute is *Hash*, which corresponds to a compressed version of the public key used for a transaction (see Sec. 2). Note that we have added also a *name* field (empty for the moment), which can be filled via other techniques [14] to also consider the real name of senders and receivers.
- *ValueTx* extends the *E*dge class (native in OrientDB): these edges connect transactions and addresses. The attributes are *Value* (the amount of bitcoins sent or received), *TIndex* (the source transaction), and *VOut* (the number of outputs).
- The *InvalidOutputScript* class. As seen in Sec. 2, an output does not always correspond to a destination address: with the OP_RETURN code in transactions it is possible to leave a message linked to a transaction. Basically, an InvalidOutputScript is an edge incident to one node only, and it is used to store such a message.

4 VISUAL ANALYTICS

Big Data analytics examines large amounts of data to uncover hidden patterns, correlations and other insights. The block-chain can be considered as Big Data: by the end of February 2017, the block-chain contains over than 200 million transactions, for more than 104,544Mb. For this reason we turned our attention to *Visual Analytics* [19] (*VA*), that is the science of analytical reasoning facilitated by interactive visual-interfaces. The main aim of VA is to help the visualisation of problems whose size, complexity, and need for closely coupled human-machine analysis may make them otherwise intractable. The goal is to rapidly visualise only the data of interest.

Being VA task-oriented [19], we have identified nine main tasks: *i)* find miners; *ii)* find transaction sources and understand how they are connected; *iii)* find the main addressees of transactions; *iv)* find the "richest" and "poorest" addresses; *v)* find the addresses with a break-even budget; *vi)* find bitcoin flows from an arbitrary address; *vii)* find bitcoin flows from a set of different addresses; *viii)* filter the block-chain on intervals of time or block identifiers; *ix)* filter the block-chain on specific transaction amounts of bitcoins, or on their number of involved addresses.

To reach such tasks, in the initial window it is possible to select among three different kinds of visualisation: *Single Transaction*, *Address Transactions*, and *Archipelago*. The first view allows for manually inserting the hash of one desired transaction, and then the tool shows the input addresses (in the following mentioned simply as "inputs") and the output addresses (in the following, "outputs") as a graph as in Fig. 3(a). The second option is the dual of the former: it is possible to type in an address and the tool shows all the transactions that have such address as output, and all the inputs of these transactions. Hence, the first two options offer a more targeted view: the user already has some initial information.

The Archipelago view is the third and most challenging one: it displays all the islands of the archipelago of Bitcoin transactions. An island is a connected component of a graph, where each couple of nodes is connected through a path, and each of the nodes is not connected to any other vertex of the super-graph. In Fig. 2(a) we show such archipelago for the block interval 1-111,111, which is made by 1,700 islands.[12] As it can be seen from Fig. 2(a), the number of islands is too large to be useful. For this reason we have created four slide-bars, each one operating on a different data-filter:

- A block-interval filter, by date or by height position (from-to) in the block-chain. Only the transactions in such blocks are visualised in the archipelago.
- A filter on the number of transactions: only the islands with the specified minimum and maximum number of transactions are shown.
- A value-based filter: it specifies the minimum and maximum amount of bitcoins considering all the transactions of a single island (i.e., their sum).

[12] For the sake of testing, at the moment only the first two years of transactions are available to the public, until 2011-02-28.

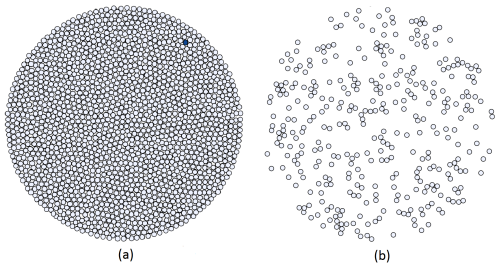

Figure 2: (a) The whole Bitcoin archipelago, and (b) by keeping only islands with 2-10 miners (min-max).

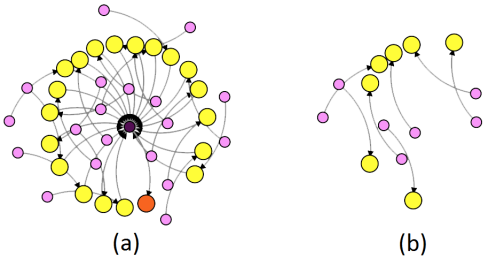

Figure 3: An example (a) of an island as visualised by BlockChainVis. Larger nodes are addresses, and smaller ones are transactions. The darker larger node has a higher (incoming) budget. In (b) we filter (a) by keeping only transactions in the first half of the block interval considered in (a).

- A filter on the number of miners: it specifies the minimum and maximum number of miners in each visualised island.

In Fig. 2(b) we lighten the visualisation w.r.t. Fig. 2(a) by only showing islands with 2-10 miners. We obtain 354 islands (20% of the total): most of the islands only have one miner.

By clicking on any island of the archipelago, a summary of its statistics pops-up. It is possible to enter into an island and visualise all its transactions. However, it is also possible to pre-filter the transactions before visualising them. The visualisation employs an oriented graph: a node can represent either a transaction or an address, and each transaction may have 1-*n* outgoing edges and 1-*n* incoming edges (see Sec. 2). The graph is bipartite: transactions can only be connected to addresses, and vice-versa. An example directly taken from BlockChainVis, is provided in Fig. 3(a). Larger nodes are addresses: they are lighter if their budget is balanced (bitcoin inputs equal to outputs). The colour of address nodes gets darker as their budget moves from balance: the sink of all the transactions in Fig. 3(a) receives 750฿ as input, without any output. Smaller nodes represent transactions, and their colour is darker if the amount of transferred bitcoins is larger.

As for the archipelago view, even the visualisation of a single island can contain too much information to be useful: three slide-bars can filter out undesired information:

- A block-interval filter by height position in the block-chain. Only transactions in such a block interval are visualised.
- A value-based filter: expressed as a min/max interval, it hides all the transaction whose value is outside this range.
- A balance-based filter: expressed as a min/max interval, it hides all the addresses whose balance (sum of the inputs minus sum of the outputs) is outside this range.

As an example, in Fig. 3 we show (a) an island and (b) the same island with only the transactions in the first half of the block interval considered for (a). Besides the previous main filters for islands, BlockChainVis has some secondary filters to *i)* show only the roots of an island, *ii)* only the leaves, *iii)* hide the transactions with a fee, *iv)* collapse binary transactions, and *v)* collapse changes: with *iv)* we hide the nodes that represent the transactions with only one input and only one output, while with *v)* we hide the transactions that return a change to the same address. The effect of some of the implemented filters is shown in Fig. 5.

Moreover, by clicking on an island in the archipelago view, a *tooltip* (or *infotip*) appears to show numerical information related to the selected island. The tooltip reports the number of transactions, the total amount of bitcoins transferred in all the considered transactions, the number of miners, the time interval between the first and the last transaction. A tooltip is also available for addresses and transactions. For a transaction the tooltip reports the hash, the timestamp, the transferred value, the fee for the miner, the number of inputs and outputs. For an address node, the information consists of its hash, its name (if any), the amount received and sent from this address, its balance, and the number of inputs and outputs. Moreover, by clicking on a transaction or an address, all the incoming and outgoing edges are automatically coloured in respectively blue and red, in order to understand the extent of their in/out-degree.

Finally, there is the possibility to select an address (or a group of them), and then to show all the paths departing from it and reaching the leafs of an island, immediately highlighting the bitcoin flows from these addresses. It is also possible to restrict only to paths with a desired number of hops (e.g., 2); this is accomplished by typing in the number of hops and then pushing the button *Show Paths*. The graph is visited through a *Breadth-first* search (*BFS*).

Model-view-controller. With respect to the MVC structure we have introduced in Sec. 3, two functions we have implemented in the Model are *getArchipelago()* and *getIsland()*, which fill the arrays of *Nodes* and *Edges*, and an array of *Links* between them, as indicated by the Controller. A *View* can be any output representation of the information contained in the Model (e.g., a chart or a diagram): two functions in our Model are *drawArchipelago()* and *drawIsland()*. Finally, the *Controller* accepts input from users and converts it to commands for the Model. In *BlockChainVis*, the Controller implements most of the functions: e.g., *findRoots()*, *highlLightMiners()*, *Bfs()*, and *Reset()*, to restore a graph to the initial unfiltered-view.

5 CASE STUDY

In this case study, we show how we have managed to spot a set of 15,964 transactions that correspond to only two flows exchanging bitcoins between two addresses. It may be possible that all such transactions belong to a mixing service, even if, due to the pseudo-anonymity of all the involved addresses, it is not possible be sure. Anyhow, this pattern is interesting to be studied due to its singularity.

Besides the licit goal of improving privacy (transactions are only pseudonymous and public), indeed such services can be also used for illegal activities, as for money-laundering. The client must provide

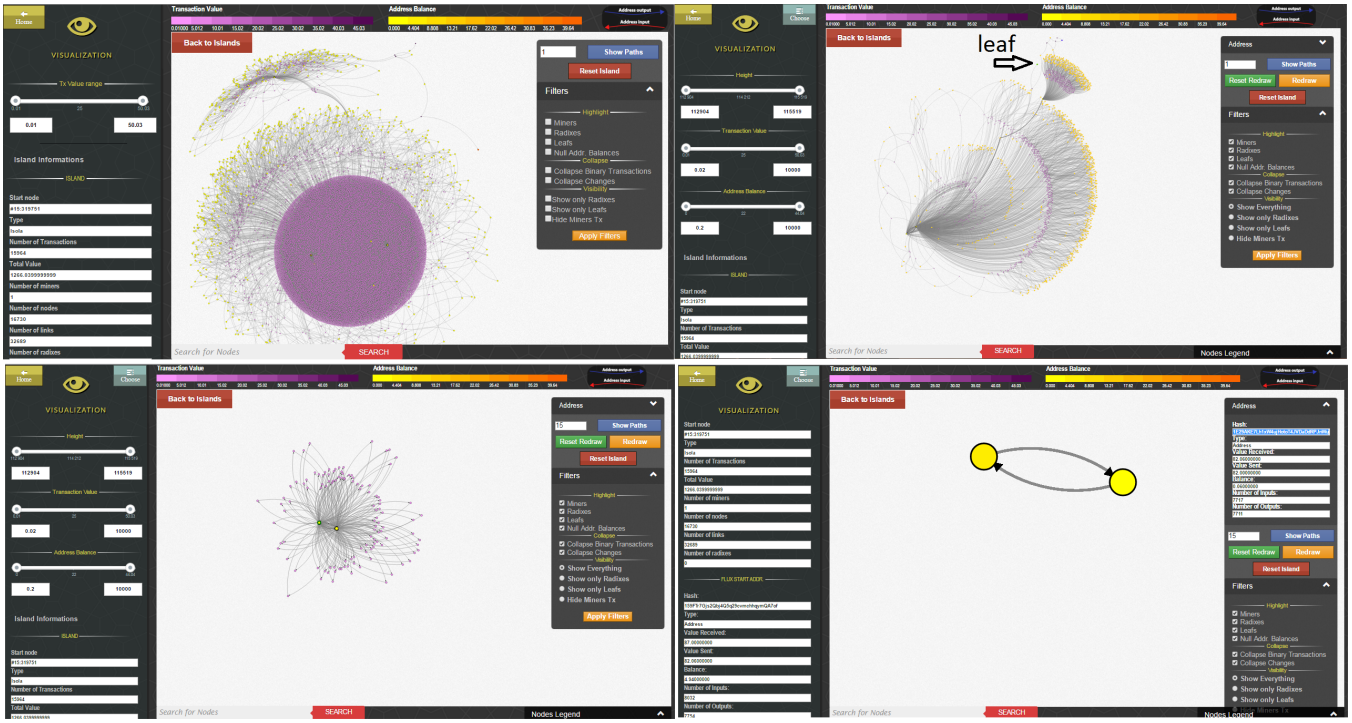

Figure 4: From the full visualisation of an island (top-left), to the only two addresses involved in all the transactions (bottom-right). First, all changes and binary transactions are collasped to discover a single leaf. Then, all the root-to-leaf paths are extracted.

the destination Bitcoin address(es) to which the mixer will deposit coins, and the transaction fee: in Bitmixer (see Sec. 1) the minimum fee is 0.5% plus 0.0005฿ for every forward address. It is often possible to set a delay before the transaction is sent. The mixer replies with a Bitcoin address of its own, selected randomly from its pool of addresses not currently involved in a transaction. The mixer also creates a record in its database mapping its Bitcoin address to the destination addresses (and further information, e.g., the time delay). Then the client sends coins to the mixer's Bitcoin address (which is not public). The mixer manipulates its wallets in such a way as to guarantee that the sent coins are taken from its existing amount of available coins, and that it does not retransmit the coins received in this transaction.

In Fig. 4 we start by considering all the information contained in an island (top-left); this image directly reports a screenshot of BlockChainVis. Indeed this screenshot is not very informative since it shows 15,964 transactions, with a total value of 1,266฿. However, by applying different filters, the situation becomes much clearer: the top-right screenshot is obtained from the first one by both collapsing binary transactions and changes (see Sec. 4), and by highlighting the miners (only one in this island), the roots, the leaves, and the addresses with a null balance. By doing this, we immediately see that the number of visualised transactions drops, and that the island contains only one leaf node. The next step consists in visualising only the paths that end to this leaf address: we select this node and we click on *Show Paths*. The result is represented in the bottom-left screenshot in Fig. 4; since we have outputs and changes still collapsed, each path is aggregated as a single transaction. If we filter out also such transactions we obtain the last screenshot in Fig. 4 (bottom-right). It is now very easy to see that all the involved transactions exchange money between only two nodes.

6 RELATED WORK

Since the block-chain is open, in combination with the networked structure of transactions, in the literature we can find several tools for visualising the Bitcoin network.

In [17] the authors present *BitIodine*; BitIodine labels users (semi) automatically with information on their identity and actions, which is scraped from openly-available information sources; for instance, individuals that a have significant negative feedback on the *BitcoinTalk*[13] trust system.

An interesting deployment of small-scale visualisation to directly analyse transactions is presented in [1]. The paper presents a tool, i.e., *BitCoinView*, which performs a bottom-up visual analysis of the influence of selected source-transactions on subsequent flows in the transaction graph. Some other approaches that permit the visualisation of part of the graph are described in [12, 14, 15]).

However, the work that is more related to our approach is however [11]. In [11] the authors present a systemic top-down visualisation of Bitcoin transaction activity to explore dynamically generated patterns of algorithmic behaviour. Differently, in our work we explicitly make use of Visual Analytics with the purpose to adopt different filters and views that allow a user to visually capture only the information of interest.

[13] BitcoinTalk: https://bitcointalk.org.

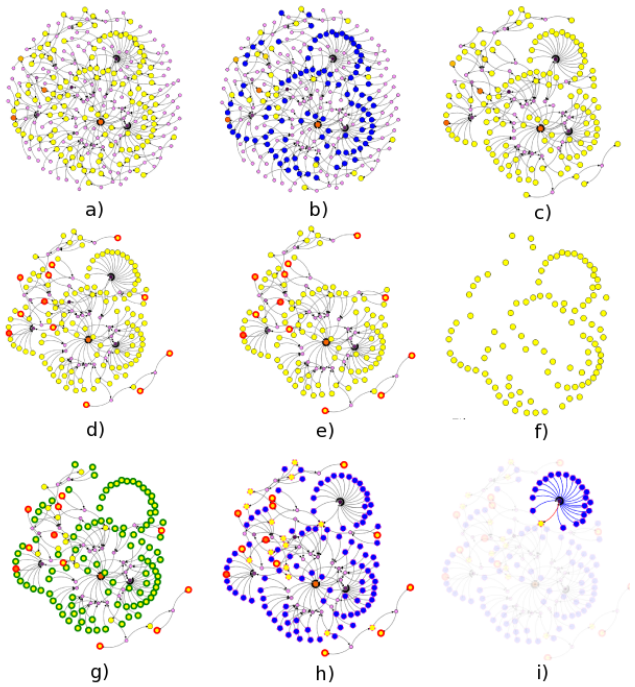

Figure 5: An example of filter application on (a) the initial graph: (b) by highlighting miners, (c) hiding coinbase transactions (rewarding the miners), (d) highlighting leaves, (e) applying a transaction-value interval, (f) showing only roots, (g) visualising paths (most of the two-colour nodes are the same roots in (f), the others are the leafs), (h) combines *b* and *f* together (darker nodes highlight the same miners in (b)), (i) focuses on a given transaction.

In addition to research works, on the Internet it is possible to find a plethora of visualisation tools. *Bitnodes* [3] shows the distribution of Bitcoin nodes across the globe, using a Bitcoin crawler implemented in Python. The live-map of *Wizbit* [18] displays both the transactions and the latest discovered Bitcoin blocks. *Interaqt* shows the live transaction-graph: the size of the nodes represents the transaction volume; every node also carries a link to its information, taken from *Bitcoin.info*[14]. *Skry* [16] is conceived for performing in-depth investigations across the block-chain. The interface can be used to analyse the relationships and to explore the transaction graph, which is enriched with addresses (possibly clustered) and other entities. *Coin Viz* [7] is a financial visualizer for Bitcoin transactions. It contains a tool, based on a force-directed approach, that shows in real-time the transactions entering the block-chain. For each of them, the sender and recipient addresses are shown. However, it is not possible to visualise all the information as we instead do in our tool (see Sec. 4). A tool with similar features is *Bitcoin-tx-graph-visualizer* [2]. *Blockseer* [6] also allows for labelling and clustering addresses by using publicly available or self-created labels. Other tools, like *Blockchain.info* [4] and *blockr.io* [5] allow a user to navigate the graph of transactions and present several charts on financial data (some of the information about Bitcoin markets used in this paper has been taken from Blockchain.info). All such tools are more oriented to showing statistical, geographical, or aggregated information, rather than offering means to focus on specific islands and to show only information filtered according to the user's needs.

[14]Bitcoin.info: http://bitcoin.info.

7 CONCLUSION AND FUTURE WORK

We have presented BlockChainVis, a visualisation tool for flows of bitcoins. The main goal is to, given a specific task, filter out not interesting information, in order to better analyse the block-chain.

In the future we would like to enhance the tool by adding some logic to automatically find the solution to specific problems: so far, it is up to the user to apply the right filters to let needed information emerge. For instance, we would like to manually select an island, and then ask to the tool if it contains a mixing service: the result will be a probability value. Moreover, we would like to make BlockChainVis more time-sensitive, in order to shown events: for instance, when money is rapidly moved from one address to another.

Finally, we are testing the use of the high performance computing and storing cloud-services provided by *Microsoft Azure*[15] and *Amazon Web Services*[16] to have more computational power.

REFERENCES

[1] Giuseppe Di Battista, Valentino Di Donato, Maurizio Patrignani, Maurizio Pizzonia, Vincenzo Roselli, and Roberto Tamassia. 2015. Bitconeview: visualization of flows in the bitcoin transaction graph. In *2015 IEEE Symposium on Visualization for Cyber Security, VizSec*. IEEE Computer Society, 1–8.
[2] Bitcoin-tx-graph-visualizer 2015. http://tinyurl.com/hpj36wd. (2015). Accessed: 2017-01-15.
[3] Bitnodes 2017. https://bitnodes.21.co. (2017). Accessed: 2017-01-15.
[4] Blockchain.info 2017. https://blockchain.info/. (2017). Accessed: 2017-01-10.
[5] Blockr.io 2017. http://blockr.io. (2017). Accessed: 2017-01-10.
[6] Blockseer 2015. https://www.blockseer.com. (2015). Accessed: 2017-01-15.
[7] Coin Viz 2017. http://tinyurl.com/jyb3dx4. (2017). Accessed: 2017-01-15.
[8] Henri Gilbert and Helena Handschuh. 2003. Security analysis of SHA-256 and sisters. In *Workshop on Selected Areas in Cryptography*. Springer, 175–193.
[9] Don Johnson, Alfred Menezes, and Scott Vanstone. 2001. The elliptic curve digital signature algorithm (ECDSA). *International Journal of Information Security* 1, 1 (2001), 36–63.
[10] Glenn E. Krasner, Stephen T. Pope, and others. 1988. A description of the model-view-controller user interface paradigm in the smalltalk-80 system. *Journal of object oriented programming* 1, 3 (1988), 26–49.
[11] Dan McGinn, David Birch, David Akroyd, Miguel Molina-Solana, Yike Guo, and William J Knottenbelt. 2016. Visualizing Dynamic Bitcoin Transaction Patterns. *Big Data* 4, 2 (2016), 109–119.
[12] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. 2013. A Fistful of Bitcoins: Characterizing Payments Among Men with No Names. In *Conference on Internet Measurement Conference (IMC '13)*. ACM, 127–140.
[13] Satoshi Nakamoto. 2008. Bitcoin: A Peer-to-Peer Electronic Cash System. http://www.hashcash.org/papers/hashcash.pdf. (2008). [Online; accessed 18-Feb-2017].
[14] Fergal Reid and Martin Harrigan. 2011. An Analysis of Anonymity in the Bitcoin System. In *PASSAT/SocialCom 2011, Privacy, Security, Risk and Trust*. IEEE, 1318–1326.
[15] Dorit Ron and Adi Shamir. 2013. Quantitative Analysis of the Full Bitcoin Transaction Graph. In *Financial Cryptography and Data Security (LNCS)*, Vol. 7859. Springer, 6–24.
[16] Skry 2017. https://skry.tech. (2017). Accessed: 2017-01-15, now acquired by http://bloq.com.
[17] Michele Spagnuolo, Federico Maggi, and Stefano Zanero. 2014. BitIodine: Extracting Intelligence from the Bitcoin Network. In *Financial Cryptography and Data Security (LNCS)*, Vol. 8437. Springer, 457–468.
[18] Wizbit 2017. https://blocks.wizb.it. (2017). Accessed: 2017-01-15.
[19] Pak Chung Wong and Jim Thomas. 2004. Visual Analytics. *IEEE Comput. Graph. Appl.* 24, 5 (2004), 20–21.

[15]https://azure.microsoft.com/en-us/.
[16]https://aws.amazon.com.